ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

MATEUS DA SILVEIRA COLISSI

TEAMUP: CONVERSATIONAL AGENTS TO SUPPORT COORDINATION IN GROUPWORK

Porto Alegre
2021

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

TEAMUP: CONVERSATIONAL AGENTS TO SUPPORT COORDINATION IN GROUPWORK

MATEUS DA SILVEIRA COLISSI

Master Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Dr. Rafael Heitor Bordini
Co-Advisor: Prof. Dr. Viviana Mascardi

Porto Alegre
2021

Ficha Catalográfica

C696t Colissi, Mateus da Silveira

Teamup : conversational agents to support coordination in groupwork / Mateus da Silveira Colissi. – 2021.
80 f.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rafael Heitor Bordini.
Co-orientadora: Profa. Dra. Viviana Mascardi.

1. Multi-Agent System. 2. JaCaMo. 3. Chatbot. 4. Dialogflow. 5. Group Coordination. I. Bordini, Rafael Heitor. II. Mascardi, Viviana. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

MATEUS DA SILVEIRA COLISSI

TEAMUP: CONVERSATIONAL AGENTS TO SUPPORT COORDINATION IN GROUPWORK

This Master Thesis has been submitted in partial fulfillment of the requirements for the degree of Master in Computer Science, of the Computer Science Graduate Program, School of Technology of the Pontifical Catholic University of Rio Grande do Sul

Sanctioned on March 24, 2021.

COMMITTEE MEMBERS:

Prof. Dr. Renata Vieira (PPGCC/PUCRS)

Prof. Dr. Jomi Fred Hübner (PGEAS/UFSC)

Prof. Dr. Viviana Mascardi (DIBRIS/UNIGE- Co-Advisor)

Prof. Dr. Rafael Heitor Bordini (PPGCC/PUCRS - Advisor)

I dedicate this work to my family.

"Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools. It is about how we use them, and what we find out when we do."
(Edsger Dijkstra)

ACKNOWLEDGMENTS

First of all, I would like to thank all my family who supported and encouraged me during the two years of the master's degree. I am also grateful for the new friendships, colleagues, laboratory colleagues and the supervisor of this work, Professor Rafael Heitor Bordini, who were always willing to aid in any matter.

I thank Professor Viviana Mascardi for agreeing to co-orient this work and Professors Renata Vieira and Jomi Hübner for agreeing to be evaluators of this work and for all contributions.

I am grateful to the Computer Science Program (PPGCC) for all the support granted during the master's degree and to UOL EdTech for the scholarship-fee.

And finally, I want to thank everyone who somehow contributed, directly or indirectly, to this work.

TEAMUP: AGENTES DE CONVERSAÇÃO PARA APOIAR A COORDENAÇÃO DE TRABALHO EM GRUPO

RESUMO

Este trabalho tem como objetivo investigar e aplicar o uso de um sistema multi-agente para auxiliar na coordenação de tarefas em grupos, especificamente em ambientes educacionais, em que a interação dos integrantes do grupo ocorre de forma indireta e não síncrona. Para uma melhor experiência do usuário, o sistema foi disponibilizado em uma interface web integrado com um chatbot para uma interação de forma mais natural. O chatbot faz a comunicação com o sistema multi-agente que é responsável pela organização do grupo, isso é, contém as informações a respeito das tarefas que devem ser realizadas e a respeito dos integrantes dos grupos, além de restrições que podem ser impostas conforme a organização de um grupo e também é capaz de retornar a informação requisitada em linguagem natural. Essa abordagem foi validada pela experiência de usuários que realizaram um curso prático de graduação em engenharia de software para testar o funcionamento e a capacidade do sistema, em que os grupos de alunos fizeram o desenvolvimento colaborativo de um software. O sistema auxilia os alunos em um projeto real desenvolvido como parte desse curso. A avaliação do sistema é realizada acompanhando as ações dos membros do grupo, através do qual podemos confirmar se o chatbot está retornando a informação correta do sistema multi-agente. Com essa avaliação, verificou-se que o sistema foi capaz de garantir integridade no desenvolvimento das tarefas dos grupos, além de garantir respostas rápidas e coerentes com a solicitação do aluno.

Palavras-Chave: Sistema Multi-Agente, JaCaMo, Chatbot, Dialogflow, Coordenação de Grupos.

TEAMUP: CONVERSATIONAL AGENTS TO SUPPORT COORDINATION IN GROUPWORK

ABSTRACT

This work aims to investigate and apply the use of a multi-agent system to assist in the coordination of tasks in groups, specifically in educational environments, in which the interaction of the members of the group occurs indirectly and asynchronously. For an improved user experience, the system was integrated into a web interface integrated with a chatbot for more natural interaction. The chatbot communicates with the multi-agent system that is responsible for the organization of the group, that is, it contains information about the tasks that must be performed and about the members of the groups, in addition to restrictions that can be imposed according to the organization of a group and is also able to return the requested information in natural language. This approach was validated by the experience of users who took a practical undergraduate course in software engineering to test the functioning and capacity of the system, in which groups of students collaboratively developed software. The system assisted students in a real project developed as part of this course. The evaluation of the system is carried out following the actions of the group members, through which we can confirm that the chatbot is returning the correct information from the multi-agent system. With this assessment, it was found that the system was able to guarantee the correct management of the group organization in the development of the group's tasks, in addition to ensuring quick and consistent responses to the students' requests.

Keywords: Multi-Agent System, JaCaMo, Chatbot, Dialogflow, Group Coordination.

LIST OF FIGURES

LIST OF TABLES

LISTINGS

LIST OF ACRONYMS

AI – Artificial Intelligence

AIML – Artificial Intelligence Markup Language

APT – Academically Productive Talk

AST – Abstract Syntax Tress

BDI – Belief-Desire-Intention

CSCL – Computer Supported Collaborative Learning

CSCW – Computer Supported Cooperative Work

LMS – Learning Management System

MAS – Multi-Agent System

MQ – Main Question

NLP – Natural Language Processing

RQ – Research Question

RST – Rhetorical Structure Theory

CONTENTS

1. INTRODUCTION

Institutions increasingly use collaboration to increase students' search for knowledge, using different types of learning methods, such as classroom learning and virtual learning. However, with different learning styles, we need different learning approaches. To assist these learning methods, several techniques and tools are being used in virtual environments, such as: chats, conversational agents and others. With the use of virtual environments, there is a concern with the learning techniques used in collaboration between people. There are several reasons for the cause of inefficiency in groups, but the main causes of inefficiencies in groups are: inefficient balance of team capacity, incorrect team dynamics, poor communication or difficult social situations [4].

According to King [34], collaborative learning may motivate studies more than individually, so in those environments, it is important to promote collaborative learning to enable group participation and interaction in a specific task, where knowledge is built through dialogues that enable the sharing of ideas and information within the group [44]. Also, we can provide important feedback for the teacher to know the interaction and discussion made by the group, as well as the individual contribution of students in problem solving [2].

In a collaborative work environment, to fulfill certain tasks / objectives, the interaction must be as a unit, coordinating actions, minimising redundant efforts, sharing resources, among others. To help coordinate the groups, management systems are being used as a solution to ensure consistency in the group's actions.

Therefore, to manage group learning is necessary to create environments that facilitate knowledge sharing and other valuable learning behaviours to help promote student discussion skills [20]. With the advancement of Artificial Intelligence (AI), there is a growing use of conversational agents such as chatbots to aid learning. However, most of the studies in this area are for individual learning.

To assist students in their collaboration, conversational agents, specifically pedagogical agents, are being used in roles such as: specialist, motivator or mentor [6]. Conversational agents are often associated as subgroups of pedagogical agents, who interact with students through natural language [32]. Conversational agents are being used to promote individual student dialogues to improve understanding of knowledge [52] and also with the potential to motivate student collaboration. Also, agents with social interaction capabilities can help in learning and idea generation by providing dynamic support for learners do a collaborative work [38, 39].

Conversational agents such as chatbots, use Natural Language Processing (NLP) techniques to gain knowledge. They must have the ability to understand context of a conversation, learn from the conversations and improve itself over time [5]. This can be achieved automatically with NLP techniques or manually providing the knowledge for a chatbot.

Natural Language Processing allows for human-machine interaction using natural languages associated with humans. Also, it is the area that investigates and analyses how the user's language text or speech inputs can be perceived and altered with computational techniques [17]. NLP techniques can be used in various applications such as machine translations, natural language processing, multilingual and cross-language information retrieval, speech recognition and Artificial Intelligent systems [14].

In a functional group, each member is responsible for one or more tasks. One of the main requirements for group work to perform well is that its members operate in harmony. Motivated to solve the problem of task resolution in educational environments, this work mainly aims to improve the coordination of groups by allowing: improve information sharing among group members; increase the group's productivity; improve performance in collaborative tasks and allow the educator responsible for the groups to be aware of the events during the performance of the tasks with the aid of a tool for the development of multi-agent systems and a chatbot.

1.1 Motivation

Collaborative learning should be presented as an important factor in student learning, since, as at present, when remote activities are being increasingly used, both as a trend and as an emergency way for education, this type of education can allow students a more comfortable and flexible way of learning. Due to this flexibility, systems that manage the activities of groups have an important role, so that, even without the face-to-face contact, the group and its responsible are fully aware of the duties and roles of the members throughout the development of work.

Therefore, the motivation of this work when using an approach with multi-agent systems, specifically with the JaCaMo multi-agent systems development platform, is due to the capacity of these systems to manage the behaviour of an organisation (group) mainly because it is possible to create and control an organisation through Moise, allowing us to to manage them in terms of association and task resolution management. In addition to testing the integration of JaCaMo with Dialogflow, in which communication with the user is carried out through a chatbot to allow a more effective and natural communication with the system. JaCaMo will be introduced in Section 2.2.1 and Dialogflow in Section 2.4.1.

1.2 Goal

This work aims to propose a solution to the problems of dynamics and communication of a group, since we assume that the groups are already well formed, with the use of a chatbot in a collaborative environment, where such agents must represent their users as part of a project group, and must assist in the organisation and communication between human users. The communication of the group members is done through the chatbot and not directly, that is, the members do not communicate directly through the system. Communication in the system did not take place between students to make the system as asynchronous as possible, thus testing the ability of requests to the chatbot and, consequently, to the multi-agent system.

This multi-agent system has the main objective of facilitating the dissemination of information about the current status of the project, allow management of multiple groups simultaneously and allowing the person in charge of the group to monitor performance during the execution of the task without disturbing the construction of knowledge, that is, there is no interruption by the system in communication with the user. In particular, the focus is on an academic environment, where it is important that members of a group including the lecturer are aware of how the project is evolving.

1.3 Main Contributions

The main contributions of this work are the results of an experiment with a real group in an educational environment using the multi-agent systems approach to assist in group coordination and the demonstration of the development of the proposed solution. We report experiences with possible problems and solutions in the use of a chatbot, perceptions are reported that must be taken into account when using a multi-agent system for the proposed situation and a systematic review focused mainly on works that use group coordination systems with the use of chatbot.

1.4 Dissertation Outline

This work is organised as follows. In Chapter 2 a basic idea of the concepts and tools that will be used in this work is presented. Chapter 3 a systematic review is performed to understand the concepts, technologies and approaches used to frighten collaborative learning. Chapter 4 the approach to resolve this proposal is presented,the system architecture, the environment available to the user, chatbot information and the multi-agent system are demonstrated. Chapter 5 presents the results and analysis of this work and finally Chapter 6 presents the conclusion of this work and discuss future work.

2. BACKGROUND

2.1 Collaborative Learning

Online collaboration allows students to take action and express their own preferences through annotations [1]. There are two approaches to improving students' overall learning ability [16, 18]: synchronous and asynchronous, in which synchronous learning activities are similar to those in the classroom, and asynchronous collaboration, in which the student receives feedback on their own results. Asynchronous online collaboration has become a more popular type of learning due to the lower cost of learning tools, minimal hardware requirements and the ability to allow students to learn at their own pace [29].

In addition, online collaboration offers the student the opportunity to absorb and question the knowledge generated by the group, while they are developing their own contribution to the tasks provided [29]. This allows us to give students a way to ask questions they want to clarify by collaborating with the group.

Asynchronous collaboration allows time to synthesise knowledge [9, 21], so learners can reflect on the subject of learning. It is a meaningful approach that provides a learning space where learners can discuss ideas in more objective and reflective ways [21]. In addition, through asynchronous learning, the teacher can receive a return of learners' level of understanding to better adapt the next material, facilitating a deeper understanding of the content [9, 25].

In synchronous collaboration, the discussion usually takes place in real time, scheduled by learners and teachers [25]. With synchronous collaboration, the group is more likely to discuss less complex issues and provide social support. Like a conversation, this approach makes the recipient more committed and motivated to read and respond to the message because a response from the recipient is expected [25].

2.2 Multi-Agent Systems

As the name says, Multi-Agent Systems (MAS) are systems made up of multiple agents. It has the capacity to build and understand a wide range of artificial social systems and can be applied in several different areas [57].

For multi-agent systems, the term autonomous designates the fact that agents have their own existence, regardless of the existence of other agents. Each agent has a set of behavioural capacities that define their competence, a set of objectives, and the necessary autonomy to use their behavioural capacities in order to achieve their objectives. The main contribution of multi-agent systems is the development of systems that are continuously running and reacting to events that characterise changes in the dynamic environments in which these autonomous systems usually operate [8].

According to Wooldridge [57], understanding a domain with multiple agents is essential to understand the type of interaction that occurs between agents. For smart self-employed agents, the ability to do business is sorely needed, negotiation and argumentation skills are often required for this.

In a multi-agent system, the collection of roles, relationships and structures of authority that govern the behaviour of agents is called organisation. Every multi-agent system has some form of organisation, even if implicit and informal. Organisations guide how agents should interact, which can influence data flows, resource allocation, authority relationships and various other system resources [24].

The study of organisational agents received a lot of attention from multi-agent researchers because several studies have shown that the organisation of a system can significantly impact their performance [24].

2.2.1 JaCaMo

JaCaMo [8] is a multi-agent systems development platform that enables integration of three multi-agent programming dimensions: agents, organisations, and environment. A JaCaMo system, as shown in Figure 2.1, consists of the following platforms: Jason [11] for agent development, CArtAgO [10] for environment programming, and Moise [27] for programming organisations. Therefore, JaCaMo integrates these three platforms for a uniform and consistent programming model, with the goal of simplifying the combination of these dimensions when programming multi-agent systems [8].

Jason [11] is a platform for multi-agent system development that incorporates an agent-oriented programming language. It makes use of AgentSpeak agent-oriented language, which is based on logical programming and the cognitive Belief-Desire-Intention (BDI) model architecture for autonomous agents. At the agent level, the main abstractions are: (i) beliefs, representing the information the agent has about other agents, the environment in which they are located and the state of the organisation; (ii) goals, representing states of what the agent would like to bring; and (iii) plans that are courses of action that can be used to achieve objectives, forming the agent's know-how [8].

CArtAgO [10] is a framework and infrastructure for environments programming and execution in multi-agent systems. It can be used as an abstraction to design multi-agent system that encapsulates functionality and services that agents can exploit at run-time [54]. These environments can be designated and programmed into artifacts. Artifacts are like a

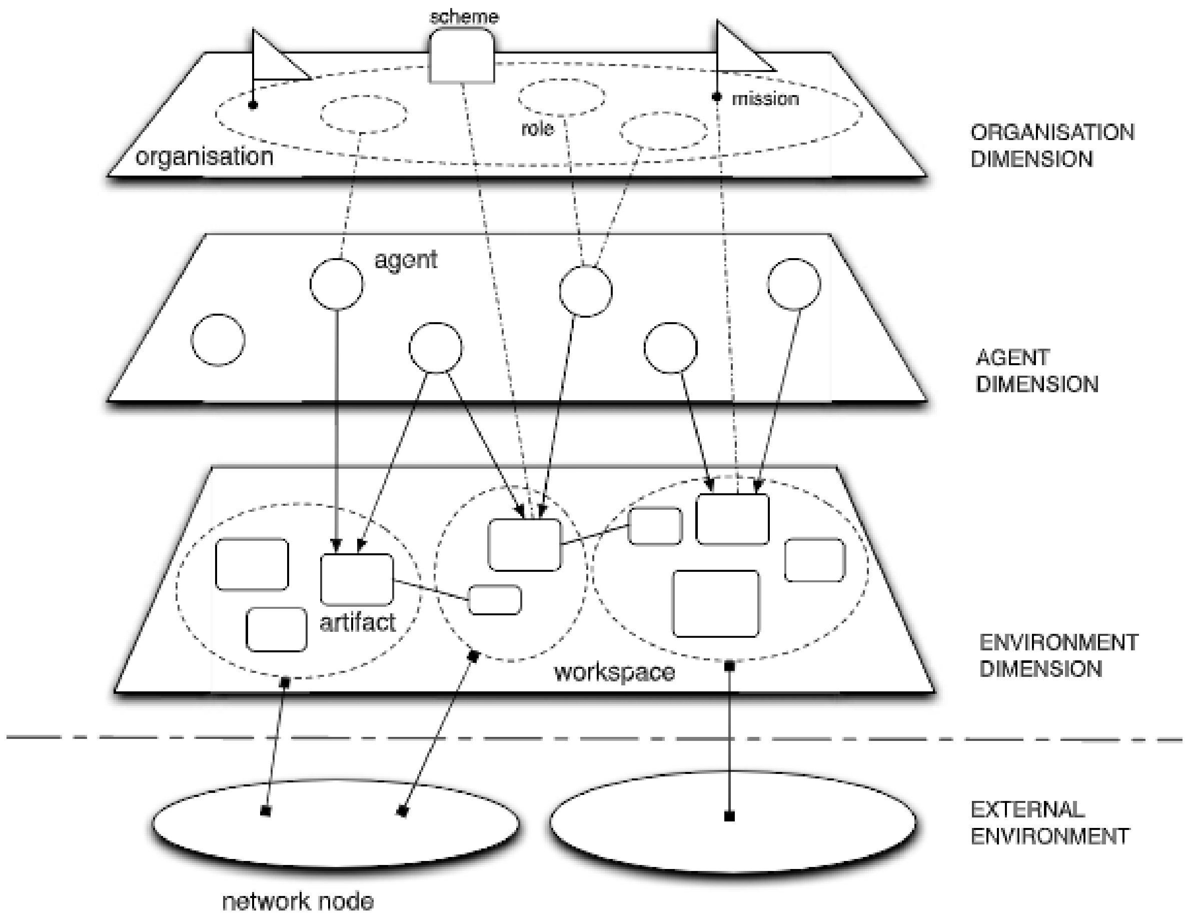


Figure 2.1 – Overview of a JaCaMo Multi-Agent System [8]

set of computational entity dynamics, collected in workspaces where the actions that agents choose to perform are performed and possibly distributed across multiple network nodes. Artifacts also have observable properties that, when altered, automatically reflect agents' beliefs through perception of the state of the environment [8].

Moise [27] implements a programming model for the organisational dimension, including: an organisation modelling language, an organisation management infrastructure [26], and support for agent-level organisation-based reasoning mechanisms. At the organisational level, agents play roles in agent groups that are jointly responsible for managing tasks. Such tasks are specified as social schemes, which break down tasks into simpler tasks. The organisation takes care of informing agents when their part of the joint task can be accomplished, thus helping to coordinate the work of various agents [8].

2.3 Self-Organising Systems

Due to the increasing complexity of the proposed tasks [47], there was a significant transformation from organised work to individual work with team-based structures, together with a focus on organisational efficiency [36]. In addition, technology innovations make it easier for members to communicate and collaborate across disparate locations.

To ensure organisation in multi-agent systems, coordination between agents is required. Coordination is related to agents' social skills, in which agents communicate with each other to share information, beliefs, goals and plans [11]. With coordination, agents can achieve joint objectives and plans that would not otherwise be possible, and ensure that tasks are performed consistently and efficiently by synchronising their actions and interactions with other agents [28, 12].

Capone et al. [13] proposed a Smart RogAgent mobile app using the JaCaMo multi-agent system, with the objective of simulating rogaining, where agents simulate human participants, groups of a JaCaMo organisation simulate teams and the environment can be simulated in a realistic way. Although its development is still in progress, there is great potential in the direction of the research.

The system architecture was: humans modelled as Jason agents, teams modelled as Moise organisations, and the resources accessed by agents modelled as CArtAgO artifacts. In the development of team building at JaCaMo, based practices were followed, such as: (i) improving the team's problem solving, managing interpersonal relationships, setting goals or clarifying roles; (ii) following up on plans/agreements to maintain accountability; and (iii) guiding the team to develop tangible action plans/agreements, must be adopted to make team building more effective.

Capone et al. [13] conjectures that the combination of argumentation scheme and ontologies brings significant new contributions to the development of practical applications of collaborative teams of human beings and autonomous agents. That is, due to non-viable reasoning patterns, which include several critical questions that can be used to justify the conclusion or avoid inference when answered negatively. These questions can make explicit reference to the roles that agents are playing and answers to those questions can also help to clarify which agent is best suited to play which role and why.

2.4 Chatbots

Chatbots are software that interacts with the user by mimicking human conversations and offers personalised services. There are two types of chatbot apps: the cloud-based chatbot that can be accessed through the web interface and a standalone chatbot app that can be accessed from a single computer [31]. Chatbot uses natural language input text and responds with the best intelligent response to user input text [53].

In order for user text entries to be identified, it is necessary to incorporate the bot with intelligence and knowledge to identify sentences and generate an appropriate response to initiate a conversation between the human and the bot. However, there is a concern with the context of the conversation, in which each platform deals differently.

There are many chatbots platforms, such as Google Dialogflow, Microsoft Azure Bot Framework and others. Dutta [17] discussed four chatbot platforms: Dialogflow, Wit.ai, Luis.ai, and Pandorabots. Dialogflow is recommended for intelligent chatbot development because it is capable of handling user input *sub-intent* goals. To create a chatbot with Dialogflow, we need to create agents and declare the *intent* streams that receive user requests in specific contexts. Data is managed through the Cloud function to access Google cloud services, enabling chatbot development without the need to create a server [46].

2.4.1 Dialogflow

It is a platform for the development of chatbots on the platform console based on natural language conversations. Chatbot model behaviour uses concepts such as *intents* and *contexts*. Where *intents* are the mapping between what a user types and what response or action should be performed by the bot, and *contexts* which are used to distinguish user entries that may have different *intent* depending on previous user entries.

Dialogflow [1] allows many integration options for our agent, providing platform specific features and creating rich responses. It can be integrated with many current conversational platforms such as Google Assistant, Slack, Facebook Messenger, Skype, Twitter, Telegram and others. Dialogflow provides agent import and export capabilities for other NLP platforms (e.g. Amazon Alexa and Microsoft Cortana). This allows us to prioritise our agent creation and let the platform handle end-user interactions.

Dialogflow has a conversation fulfillment (deployed as a web-hook) that calls a REST API or back-end service such as C#, Go, Java, Node.js, PHP, Python and Ruby to return a response to conversation platforms. In addition, it can handle Dialogflow *intents* individually, allowing to specify which *intent* will use extra processing and which will return only defined responses.

When receiving a user input, it is first checked to match a predefined *intent*. If user entries do not match any predefined *intent*, the *default fallback intent* will handle the entry. Matching cases of an *intention* can be limited by stating a list of *contexts* that should be working and that can create and delete *contexts*.

Intents and *contexts* can provide a large and complex flow in human-computer conversation in chatbot development. However Dialogflow cannot be designed so that an *intent* can be matched only if a specific context is not present, making this a platform limitation.

[1] Available in https://dialogflow.com/

2.4.2 Wit.ai

Wit.ai [2] develops chatbots in platform console, were the training of NLP engine use examples and allows integration with Node.js, Python, Ruby and others HTTP API.

Wit.ai use the concept of stories. Each story is depicted as an example of a conversation. The *intent* is a user *entity* which is not mandatory. To create a complex chatbot a large number of *intent* in stories are grouped. When a user writes a request of similar nature, the *entities* are extracted and the logic implemented by the developer is applied.

The platform documentation is divided into three parts: understand, were the focus is categorisation, extraction and analysis; integrate Wit everywhere, where it shows how to integrate the platform with applications; and manage wit apps. Unlike Dialogflow, we don't have specifications of fulfillment or detailed description of a subject like *intents*.

2.4.3 Luis.ai

The platform allows the user to build their own model of chatbot. Example phrases or *utterances* are supplied for the *intents*. *Utterance* sentence or speech, is user's input which is supposed to be understood by the chatbot that can be labelled in Luis [3] application with specific details. Labels have two purposes: illustrate the performance of the current model on unseen data and also can be used to act as a rotating test set; and can act as an accelerator if the proposed labels are correct [56].

In Luis an *intent* is a goal conveyed in a user's input that represents relevant detailed information in the *utterance* that may be mapped to many *utterance* variations and describe user actions that is expected to perform.

Luis does not work with context, to do that, we must define context in our bot by using *dialogs*. *Dialog* works like a function in a program, it is designed to perform a specific operation an it can be invoked as often as it is needed. To handle conversation flow we need to chain multiple *dialogs*.

2.4.4 Pandorabots

The platform is based on AIML (Artificial Intelligence Markup Language). The purpose of the Pandorabots [4] is to enable human computer conversation without considering a

[2]Available in https://wit.ai/
[3]Available in https://www.luis.ai/home
[4]Available in https://home.pandorabots.com/home.html

task or action-oriented scenario. The application is an XML-based platform and conversational patterns. It can take much effort to scale up, if the application are built manually.

Platform documentation is sparse. There are only tasks to assist in learning AIML language, API references, deploy and extend application. Another problem is that sandbox version from Pandorabots, that is the free version, does not support API access.

2.4.5 IBM Watson

Watson Assistant [5] develops the chatbots in platform console using concepts like *intents*, *dialogs* and *entities* runs on a powerful cloud service (IBM Cloud) delivering a robust and interactive experience through API endpoints. Watson provides natural language understanding (e.g. speech to text) and also have a wide range of Watson services that can be useful for creating chatbots, virtual assistants and conversational agents due to run on cloud.

Watson's *intents* foresee users' goals with samples of intents to deal with users input. To create a flow of conversation is used *dialogs* to incorporate the *intents* and to create context we use *entities*.

2.5 Team Composition

Although it was not applied in this work, solving the problem of team composition is an important task for future work, as it is an essential task to allow the best collaboration of the students. In order to prevent teams from working less efficiently, team composition is a problem that has aroused the interest of research in groups, such as multi-agent systems. Solving the team composition problem for virtual teams can improve team members' relationships under their work environment remotely, enabling them to complete tasks on a daily basis, improving collaboration, productivity and tracking tasks.

The question of team composition is how to create a multi-agent system environment as a group of heterogeneous agents (e.g. humans and robots) and how to organise their activities. Team members must observe the environment and interact with each other to perform tasks or solve problems that are beyond their individual capabilities. The algorithms for creating these teams are inspired by human teamwork [3].

Recent algorithms are inspired by human teamwork because most approaches represent agent resources in a Boolean manner (whether or not an agent has a required

[5] Available in https://www.ibm.com/watson

skill). This approach does not correspond to real life, resources are not binary as each individual shows different performances for each competency [4].

Andrejczuk et al. [4] propose an approach to synergistic teams, in which the synergistic values are capacity and personality, i.e. each team is proficient (covers the skills needed for a given task) and congenial (balances gender and psychological traits). To determine personality, the author uses the method of Post-Jungian Personality Theory [55].

The Post-Jungian Personality Theory method is a modified version of the Myers-Briggs Type Indicator (MBTI) [43] that uses numerical data collected using the questionnaire. Compared to other methods, the personality questionnaire is short, containing only 21 quick questions.

The theory is based on the cognitive-mode personality model of psychiatrist Carl Gustav Jung [30]. His model has two sets of pairs of human personality variables: psychological functions and psychological attitudes.

The variables of psychological function are: (i) sensing/intuition (SN), which describes how to approach problems; and (ii) thinking/feeling (TF), which describes how to make decisions. The variables of psychological attitudes are: (i) perception/judgement (PJ), which describes the way of living; and (ii) extroversion/introversion (IS), which describes the way of interacting with the world. For example, thinking/feeling (TF), a value between -1 and 0 means that a person is sentimental and a value between 0 and 1 means that he is thoughtful.

To solve the team's synergistic composition, agents are divided into partitions and assigned competencies in a task to team members. Since the goal is to find the most competently and psychologically balanced team, the Bernoulli-Nash function is used to measure the synergistic value of a team partition [43].

Finally, the algorithm was evaluated in the context of a classroom. The results showed that the approach is better able to predict team performance than experts who know students, their social background, skills and cognitive abilities [4]. In addition, the author states that the algorithm is potentially useful for any organisation facing the need to optimise its troubleshooting teams.

Farhangian et al. [19] propose a performance computation mechanism for software development projects by taking into consideration employees' personalities and skills. The model is based on MBTI and Belbin Team Roles (BTR) [7] studies. Belbin introduced a theory about the roles of individuals in teams, that in each team each member has a role that can affect team performance and suggesting that personality and role trends are not independent. Also suggests two main factors for forming a team: dyadic relationships of team members and competency of team members in the tasks [7].

Farhangian et al. [19] use the formal model to select the best team composition for a given task. To calculate the performance of each team composition the formula used is described below:

$$\begin{aligned} \mathit{Performance} = (c1 * \mathit{Pm} + c2 * \mathit{Rm} + c3 * \mathit{Cr} + c4 * \mathit{Um} + c5 * \mathit{So} + c6 * \mathit{Co} + \\ c7 * \mathit{Bcr} + c8 * \mathit{Bum} + c9 * \mathit{Bso} + c10 * \mathit{Bco}) * c11 * C \end{aligned} \tag{2.1}$$

The above parameters are: Matching_index (*Pm*), Matching_roles (*Rm*), Creativity (*Cr*), Urgency (*Um*), Sociality (*So*), Complexity (*Co*) and Competence (*C*), parameters that starts with *B* (e.g. *Bum*) are Belbin parameters. These variables are numerical values that can be uniformly considered to be measured over some scale from 0 to 1. Identifiers *c* are coefficients that can adjust empirical measurements. For each parameter there is a formula and associated task styles such as: Creativity (*Cr*) for tasks requiring a high level of creativity, teams composed of differing attitude tendencies are believed to perform better than teams of like-minded people.

A agent based model for task allocation was developed with a system that looks for all possible combinations of a team and calculates the highest value coalition. The system assigns tasks to employees to maximise system utility. To analyse the model some simulation experiments was conducted on the NetLogo [50] platform. The simulation environment can provide a low cost simulation to investigate the impact of agent attributes, tasks, environment dynamics, and also their task allocation strategies on team performance [19]. The results presented by the author are a modelling and simulation approach that can demonstrate interesting effects based on combinations of personality and skill setting parameters. Parameterisation can also be configured for situations with specific contexts.

In multi-agent systems there is also the formation of coalitions, in which there are several agents interacting in the same environment, performing their own tasks, but that sometimes a task may require more effort than another agent can offer. The formation of coalitions, that is, the group of agents that will work together, studies how the problem of grouping these agents is approached in order to maximise the rewards they receive for their efforts [37].

Sandholm et al. [48] presented the process of how the agents are grouped and what their purpose is. The coalition formation process was divided into three activities: generation of the coalition structure; resolution of the optimisation problem of each coalition; and division of the solution value among the agents.

Wooldridge [57] referred to these activities as the life cycle of cooperation. It is shown how agents, brought together on the basis of some criteria, can cooperate with each other to achieve their desired goals. From single agents, the coalition begins to decide which agent will be responsible for which task, and then a plan is created to meet the goal.

3. RELATED WORK

A systematic literature review help us to identifying, evaluating and interpreting all available research relevant to our research area [35]. To draw the basis of a hypothesis from a research is identified and reported researches that support or not the initial hypothesis. Approaches such as, systematic or mapping reviews in bibliographic research can be used to examine where empirical evidence supports or contradicts theoretical hypotheses, or even to help generate new hypotheses [35]. We therefore chose to conduct a systematic review to identify: existing studies on collaborative learning using chatbots in education, research gaps and possible new research activities to support our hypothesis that management systems are capable of maintaining an organisation and control of collaborative work in educational environments.

3.1 Research Questions

We created the main question (MQ) and five research questions (RQs) in our systematic review to determine how collaborative work is addressed in the studies:

(MQ) How chatbots are approached with organising systems in collaborative work in education?

- **(RQ1)** What technologies are used?
- **(RQ2)** Which activities are used: individual, collaborative (how many people), which formats (practical or playful)?
- **(RQ3)** What are the advantages of collaborative activities?
- **(RQ4)** How the proposal was validated?
- **(RQ5)** What are the challenges and limitations of the proposed solution?

3.2 Search Strategy

To define our search strategy in the systematic review, we first created a preliminary research question to manually select our control paper. The control paper is selected to validate the automated search ensuring relevance of the search result. Having defined our control paper we extracted two main keywords for our research, where they are: chatbot and collaborative learning. In addition, we add the synonyms for both keywords: chat, virtual assistant, chatterbot, bot, conversational agent, CSCW, computer supported cooperative work, CSCL, computer supported collaborative learning, learning methods, group assignment, group course work, group exercise and group project.

With the previous search scope, our search string is formulated and an automated search is made in know databases such as: Scopus[1], ACM Portal[2] and IEEE Xplore[3].

3.3 Search String

The search strings used are described below. Searches were performed on OCTOBER 23, 2019 and JUNE 09, 2020 in the abstract, title and keywords fields, with publication year after 2013. We restrict year of publication due to studies using chatbot and recent NLP techniques:

Scopus

```
TITLE-ABS-KEY (("Chatbot" OR "Chat" OR "Virtual assistant" OR "Chatterbot" OR "Bot"
OR "Conversational agent" ) AND ( "Collaborative Learning" OR "CSCW" OR "Computer
Supported Cooperative Work" OR “CSCL” OR “Computer Supported Cooperative Learning”
OR "Learning methods" OR "Group assignment" OR "Group coursework" OR "Group
exercise" OR "Group project"))
```

ACM

```
acmdlTitle:(("Chatbot" OR "Chat" OR "Virtual assistant" OR "Chatterbot" OR "Bot" OR
"Conversational agent") AND ("Collaborative Learning" OR "CSCW" OR "Computer
Supported Cooperative Work" OR "CSCL" OR "Computer Supported Cooperative
Learning" OR "Learning methods" OR "Group assignment" OR "Group coursework" OR
"Group exercise" OR "Group project")) OR
recordAbstract:(("Chatbot" OR "Chat" OR "Virtual assistant" OR "Chatterbot" OR "Bot" OR
"Conversational agent") AND ("Collaborative Learning" OR "CSCW" OR "Computer
Supported Cooperative Work" OR "CSCL" OR "Computer Supported Cooperative
Learning" OR "Learning methods" OR "Group assignment" OR "Group coursework" OR
"Group exercise" OR "Group project")) OR
keywords.author.keyword:("Chatbot" OR "Chat" OR "Virtual assistant" OR "Chatterbot" OR
"Bot" OR "Conversational agent") AND ("Collaborative Learning" OR "CSCW" OR
"Computer Supported Cooperative Work" OR "CSCL" OR "Computer Supported
Cooperative Learning" OR "Learning methods" OR "Group assignment" OR "Group
coursework" OR "Group exercise" OR "Group project"))
```

[1] https://www.scopus.com/search/form.uri?display=basic
[2] https://dl.acm.org/
[3] https://ieeexplore.ieee.org/Xplore/home.jsp

IEEE Xplorer

```
(("Document Title":"Chatbot" OR "Document Title":"Chat" OR "Document Title":"Virtual assistant" OR "Document Title":"Chatterbot" OR "Document Title":"Bot" OR "Document Title":"Conversational agent") AND
("Document Title":"Collaborative Learning" OR "Document Title":"CSCW" OR "Document Title":"Computer Supported Cooperative Work" OR “Document Title”:“CSCL” OR “Document Title”:“Computer Supported Cooperative Learning” OR "Document Title":"Learning methods" OR "Document Title":"Group Assignment" OR "Document Title":"Group coursework" OR "Document Title":“Group exercise” OR "Document Title":“Group project”)) OR
(("Abstract":"Chatbot" OR "Abstract":"Chat" OR "Abstract":"Virtual assistant" OR "Abstract":"Chatterbot" OR "Abstract":"Bot" OR "Abstract":"Conversational agent") AND
("Abstract":"Collaborative Learning" OR "Abstract":"CSCW" OR "Abstract":"Computer Supported Cooperative Work" OR “Abstract”:“CSCL” OR “Abstract”:“Computer Supported Cooperative Learning” OR "Abstract":"Learning methods" OR "Abstract":"Group Assignment" OR "Abstract":"Group coursework" OR "Abstract":“Group exercise” OR "Abstract":“Group project”)) OR
(("Author Keywords":"Chatbot" OR "Author Keywords":"Chat" OR "Author Keywords":"Virtual assistant" OR "Author Keywords":"Chatterbot" OR "Author Keywords":"Bot" OR "Author Keywords":"Conversational agent") AND
("Author Keywords":"Collaborative Learning" OR "Author Keywords":"CSCW" OR "Author Keywords":"Computer Supported Cooperative Work" OR "Author Keywords":"CSCL" OR "Author Keywords":"Computer Supported Cooperative Learning" OR "Author Keywords":"Learning methods" OR "Author Keywords":"Group Assignment" OR "Author Keywords":"Group coursework" OR "Author Keywords":“Group exercise” OR "Author Keywords":“Group project”))
```

3.4 Study Selection

To filter automated search results, inclusion (I) and exclusion (E) criteria have been defined to accept or reject the study in the next phase of the review, which are:

(I1) Must have “Collaborative work in education” in the title, abstract or keywords;

(E1) Not be in English;

(E2) Not related to the theme of “Collaborative work in education”;

(E3) Repeat studies (will be considered the most recent);

(E4) Books and abstracts from conference presentations;

(E5) Narrative reviews, comparative studies, surveys, and other systematic reviews.

3.5 Control Paper

For this review we defined a control paper that had been previously identified in non-systematic searches in the Scopus database to validate the search string. The selected article is: Design of a collaborative learning environment integrating emotions and virtual assistants (Chatbots) [15]. This paper was chosen because it presented a collaborative environment in an educational context, in addition to using a chatbot to improve the productivity of groups in learning.

3.6 Execution

This section shows how the articles were selected, the selection procedure and the results were extracted. The selection of papers was performed as follows:

1. String execution on each search base;
2. Applying the filter in the bases to select papers after 2013;
3. Export of paper list in *bibtex* format;
4. Importing *bibtex* files into the *Start* [4] tool, which is a tool that supports the organisation of systematic reviews;
5. A filtering of the papers was performed, in which the selection criteria were applied based on the reading of the abstract, keywords and title, for the 609 papers returned from the different bases;
6. Full reading of previously accepted articles, in which 12 papers were accepted. Table 3.1 shows the number of articles found, duplicated and accepted for each database.

Table 3.1 – Papers returned in search

Base	**Number of papers returned**	**Number of duplicate papers**	**Number of articles accepted**
Scopus	308	38	5
ACM Portal	265	2	2
IEEE Xplore	36	5	5
Total	609	45	12

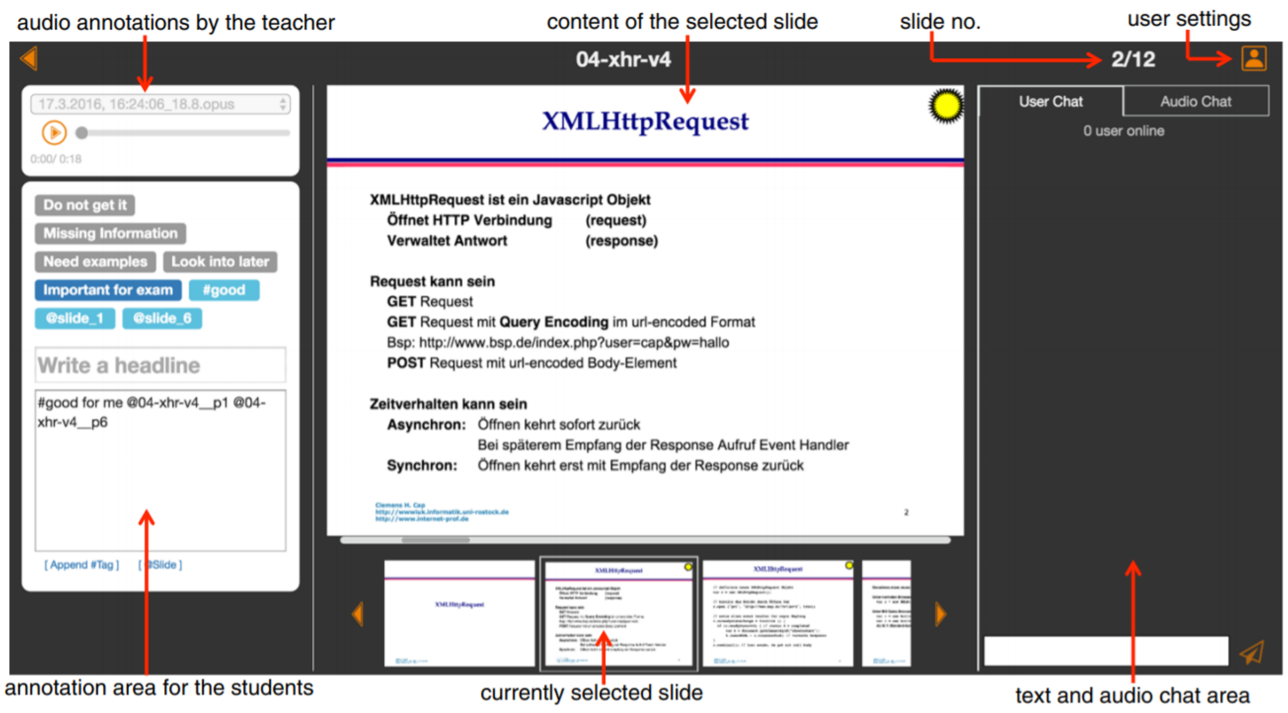
[4]http://lapes.dc.ufscar.br/tools/start_tool

3.7 Results

We set out research questions in Section 3.1, which defined the purpose of the systematic review. This chapter presents the results and analysis of the studies, addressing each of these research questions following the execution of the Section 3.6 and manually selected studies that were useful for the construction of this work. Although other works of the review were cited, in the selection of articles, only those who somehow had a group coordination in an educational context, that is, who did not use the chatbot just to extract information, were accepted in the selection.

(RQ1) What technologies are used?

Islam et al. [29] proposed a browser-based client interface approach with Express and SocketIO framework, Node.js, Jade and AngularJS, that can be accessed using only a web browser available on PCs, tablets, laptops or mobile devices, where the teacher uploads the lecture to the site for students to study/solve by collaborating in pairs as shown in Figure 3.1.



#good for me @04-xhr-v4__p1 @04-xhr-v4__p6

XMLHttpRequest

XMLHttpRequest ist ein Javascript Objekt
Öffnet HTTP Verbindung (request)
Verwaltet Antwort (response)

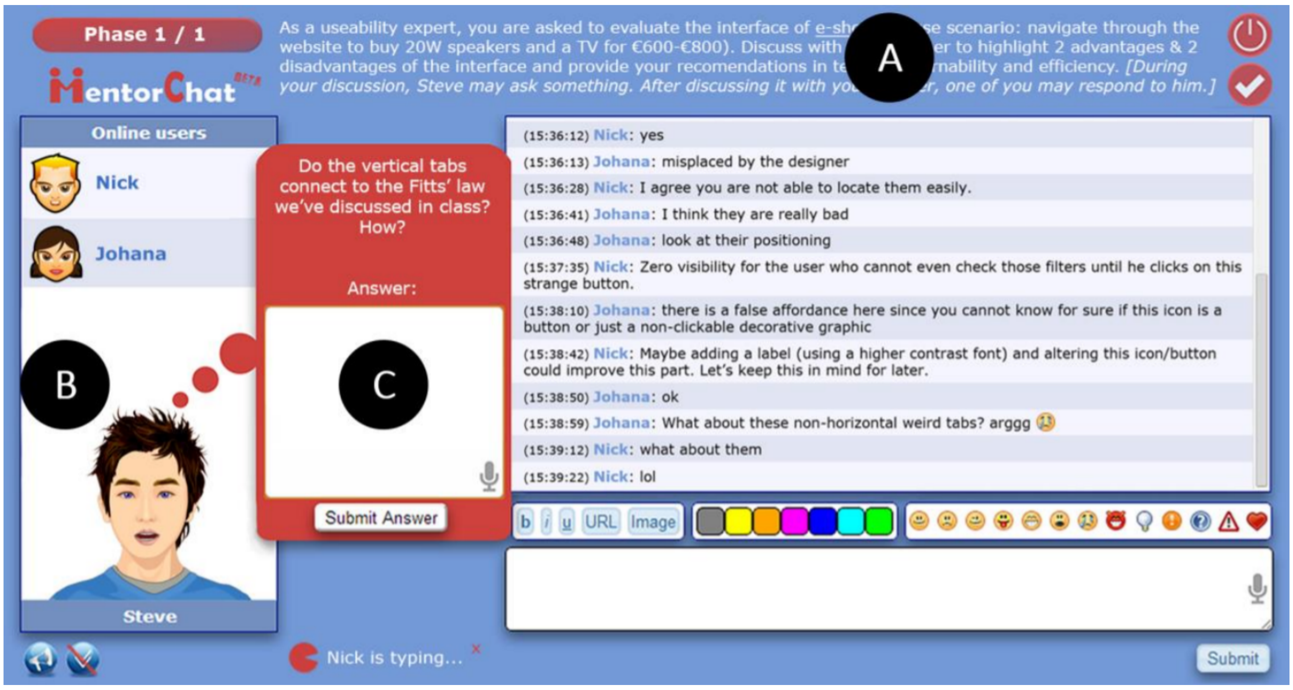
Request kann sein
GET Request
GET Request mit **Query Encoding** im url-encoded Format
Bsp: http://www.bsp.de/index.php?user=cap&pw=hallo
POST Request mit url-encoded Body-Element

Zeitverhalten kann sein
Asynchron: Öffnen kehrt sofort zurück
Bei späterem Empfang der Response Aufruf Event Handler
Synchron: Öffnen kehrt erst mit Empfang der Response zurück

Clemens H. Cap
http://wwwiuk.informatik.uni-rostock.de
http://www.internet-prof.de
2

Figure 3.1 – Collaboration window [29]

Tegos and Demetriadis [49] proposed a prototype dialogue support system, acting as an instant messaging application, in order to accomplish one or more learning tasks in an online activity (Figure 3.2A), with a conversation agent (Figure 3.2B) that uses text-to-speech to read its interventions, offering Academically Productive Talk (APT). To encourage collaboration, the agent introduces concepts into the group discussion displaying outside the main chat frame (Figure 3.2C) using the Levenshtein-based string similarity algorithm and a WordNet [42] lexicon for synonyms to calculate a proximity score for each identified concept.



As a useability expert, you are asked to evaluate the interface of e-sh se scenario: navigate through the website to buy 20W speakers and a TV for €600-€800). Discuss with er to highlight 2 advantages & 2 disadvantages of the interface and provide your recomendations in te nability and efficiency. *[During your discussion, Steve may ask something. After discussing it with yo r, one of you may respond to him.]*

Do the vertical tabs connect to the Fitts' law we've discussed in class? How?

(15:36:12) Nick: yes
(15:36:13) Johana: misplaced by the designer
(15:36:28) Nick: I agree you are not able to locate them easily.
(15:36:41) Johana: I think they are really bad
(15:36:48) Johana: look at their positioning
(15:37:35) Nick: Zero visibility for the user who cannot even check those filters until he clicks on this strange button.
(15:38:10) Johana: there is a false affordance here since you cannot know for sure if this icon is a button or just a non-clickable decorative graphic
(15:38:42) Nick: Maybe adding a label (using a higher contrast font) and altering this icon/button could improve this part. Let's keep this in mind for later.
(15:38:50) Johana: ok
(15:38:59) Johana: What about these non-horizontal weird tabs? arggg
(15:39:12) Nick: what about them
(15:39:22) Nick: lol

Figure 3.2 – Learning environment [49]

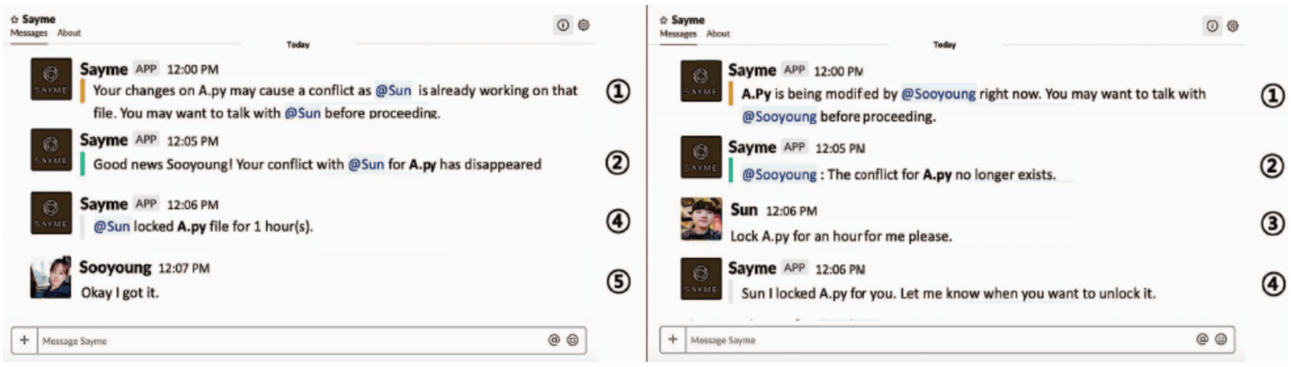
David et al. [15] uses the Alexa virtual assistant along with capturing the six basic emotions: joy, anger, sadness and fear captured by camera and text with emojis. The author proposes an approach that students with major problem solving problems and showing negative emotions will be guided first by the teacher. Learners with satisfactory task resolution results and positive emotions during the process can continue to be supervised by the virtual assistant [15].

Neto and Fernandes [44] proposed an interaction between chatbot and Learning Management System (LMS) participants to encourage discussion in the educational context to support the process of a academically productive talk.

Guo et al. [23] designed a prototype system with WebSockets, Node.js, modified Mozilla's TogheterJS library and integrated it with the Online Python Tutor to promote human interaction in multi-user program visualisations for real-time tutoring and collaborative learning.

Allaymoun and Trausan-Matu [2] proposed a model for analysing Computer Supported Collaborative Learning (CSCL) chat collaboration using rhetorical schemes to enable teachers to evaluate semiautomatic chats with NLP tools.

Paikari et al. [45] proposed a chatbot called Sayme to address the detection and resolution of of possible code conflicts that may arise in the development of parallel software as shown in Figure 3.3. Sayme is implemented using Python, MySQL, Google Cloud Platform and Slack is used as the chat channel to communicate with developers. In addition, the chatbot operates autonomously, initiating conversations with developers based on information collected from Git. Sayme also monitors when developers save files using Git commands to automatically extract files that are being changed and on which lines they are being changed. To detect possible indirect conflicts between files, all files are analysed using the Abstract Syntax Tress (AST) library.



Sayme APP 12:00 PM
Your changes on A.py may cause a conflict as @Sun is already working on that file. You may want to talk with @Sun before proceeding. ①

Sayme APP 12:05 PM
Good news Sooyoung! Your conflict with @Sun for **A.py** has disappeared ②

Sayme APP 12:06 PM
@Sun locked **A.py** file for 1 hour(s). ④

Sooyoung 12:07 PM
Okay I got it. ⑤

Sayme APP 12:00 PM
A.Py is being modifed by @Sooyoung right now. You may want to talk with @Sooyoung before proceeding. ①

Sayme APP 12:05 PM
@Sooyoung : The conflict for **A.py** no longer exists. ②

Sun 12:06 PM
Lock A.py for an hour for me please. ③

Sayme APP 12:06 PM
Sun I locked A.py for you. Let me know when you want to unlock it. ④

Figure 3.3 – Example of interaction with Sayme (direct conflict) [45]

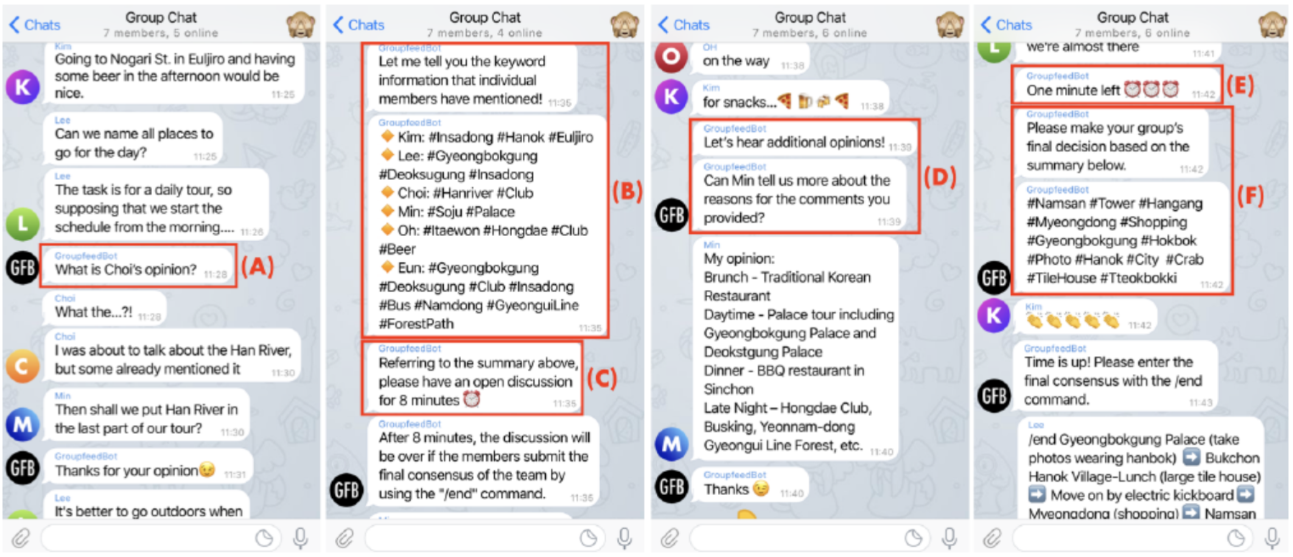
Sayme provides information about potential direct and indirect conflicts, helping developers to resolve those conflicts. Its main function is to proactively detect possible conflicts between developers working in parallel, notifying them before the conflicts become too big. Its secondary function is to respond to a variety of requests that help developers understand the state of work of other developers.

The chatbot operates on a request-response basis for its reactive features, waiting for a developer to start a conversation. Is not limited to a fixed set of commands that it would recognise on the Slack channel, for this purpose natural language processing is used to interpret the developer's request. Dialogflow is used as the underlying mechanism, with 28 to 54 common training phrases by which developers can interact with Sayme during the initial tests.

Kim et al. [33] proposed GroupfeedBot, a chatbot agent to facilitate group discussions in social chat groups on goal-oriented communication, managing discussion time, encouraging members to participate evenly and organising members' opinions as shown in Figure 3.4. The chatbot runs in the Telegram messaging application and was built with BotFather. The back-end server was created with Python, using the Telegram library and pickleDB. The front-end and back-end use a Telegram dispatcher to communicate, transmit data and access APIs.

The chatbot manages discussion time (Figure 3.4C e Figure 3.4E), facilitates participation by encouraging less active agents to speak up (Figure 3.4A e Figure 3.4D) and organises the opinions of individual members (Figure 3.4B) and general groups (Figure 3.4F).

To encourage uniform contributions, non-participating members are encouraged. Members who have not commented are subject to the following question: "What is <member name> opinion?" (Figure 3.4A), in addition, comments are also requested from less frank



Kim
Going to Nogari St. in Euljiro and having some beer in the afternoon would be nice. 11:25

Lee
Can we name all places to go for the day? 11:25

Lee
The task is for a daily tour, so supposing that we start the schedule from the morning.... 11:26

GroupfeedBot
What is Choi's opinion? 11:28

Choi
What the...?! 11:28

Choi
I was about to talk about the Han River, but some already mentioned it 11:30

Min
Then shall we put Han River in the last part of our tour? 11:30

GroupfeedBot
Thanks for your opinion😉 11:31

Lee
It's better to go outdoors when

GroupfeedBot
Let me tell you the keyword information that individual members have mentioned! 11:35

GroupfeedBot
🔶 Kim: #Insadong #Hanok #Euljiro
🔶 Lee: #Gyeongbokgung #Deoksugung #Insadong
🔶 Choi: #Hanriver #Club
🔶 Min: #Soju #Palace
🔶 Oh: #Itaewon #Hongdae #Club #Beer
🔶 Eun: #Gyeongbokgung #Deoksugung #Club #Insadong #Bus #Namdong #GyeonguiLine #ForestPath 11:35

GroupfeedBot
Referring to the summary above, please have an open discussion for 8 minutes ⏰ 11:35

GroupfeedBot
After 8 minutes, the discussion will be over if the members submit the final consensus of the team by using the "/end" command. 11:35

OH
on the way 11:38

Kim
for snacks...🍕🍺🍻🍕 11:38

GroupfeedBot
Let's hear additional opinions! 11:39

GroupfeedBot
Can Min tell us more about the reasons for the comments you provided? 11:39

Min
My opinion:
Brunch - Traditional Korean Restaurant
Daytime - Palace tour including Gyeongbokgung Palace and Deokstgung Palace
Dinner - BBQ restaurant in Sinchon
Late Night – Hongdae Club, Busking, Yeonnam-dong Gyeongui Line Forest, etc. 11:40

GroupfeedBot
Thanks 😉 11:40

we're almost there 11:41

GroupfeedBot
One minute left ⏰⏰⏰ 11:42

GroupfeedBot
Please make your group's final decision based on the summary below. 11:42

GroupfeedBot
#Namsan #Tower #Hangang #Myeongdong #Shopping #Gyeongbokgung #Hokbok #Photo #Hanok #City #Crab #TileHouse #Tteokbokki 11:42

Kim
👏👏👏👏👏 11:42

GroupfeedBot
Time is up! Please enter the final consensus with the /end command. 11:43

Lee
/end Gyeongbokgung Palace (take photos wearing hanbok) ➡️ Bukchon Hanok Village–Lunch (large tile house) ➡️ Move on by electric kickboard ➡️ Myeongdong (shopping) ➡️ Namsan

Figure 3.4 – Group chat discussions [33]

members. The number of words each member sent is counted, and then asks the person with the least participation to elaborate their opinions with the question: “Can <member name> tell us more about the reasons for the comments you provided?" (Figure 3.4D).

In addition, the chatbot establishes time constraints and alerts (Figure 3.4E) for each task to reach consensus efficiently (Figure 3.4C). Also, the main opinions of each member are organised in the form of a hashtag (Figure 3.4B), using a text classification algorithm to summarise the comments of each member in one or two sentences, thus extracting their lexical morphemes. Finally, the comments of the entire discussion are summarised (Figure 3.4F) and filtered by a text classification algorithm to transform the entire team's production into four to five sentences; with this, the main morphemes are extracted from the summarised sentences and presented as hashtags.

Göschlberger and Brandstetter [22] proposed an information system for corporate e-learning, providing learning resources through a chat interface with Google's Dialogflow as Conversational AI platform. It addresses the gap of dynamic role assignments by proposing information including providers and consumers, that is, the design objectives are constructed from both perspectives. In addition, possible regulatory demands or restrictions are also identified.

The provider, in e-learning context, is responsible for making content available online, that is, adapting, organising, creating, orchestrating and providing learning resources. In the corporate context, most of the learning resources are the existing material transformed into learning resources through learning management systems. The consumer, in the context of e-learning, is a student. In the corporate context, learning is often incorporated (or integrated) in daily work.

The system architecture consists of several services that are communicating through different REST endpoints. Postgres are used to read/write information in the relational database, the Elasticsearch search engine and a fulfillment for customised responses for a given intent.

For the conversational model, seven intents were trained: (i) Welcome: to greet users; (ii) Discovery Topics: to return all available topics in the database; (iii) Search: searches for a term passed as a parameter to process in the back-end; (iv) Learn: starts the learning process, this intention needs a topic for research and will use padding if the parameter is not provided; (v) Continue Learning: it can only be used when the context is about learning to return the next information about the current topic; (vi) Cancel Learning: it can only be used when the context is about learning and removes the learning context from the current conversation; and (vii) Fallback: when it does not correspond to any intention mentioned.

(RQ2) Which learning activities are used: individual, collaborative (how many people), which formats (practical or playful)?

Islam et al. [29] proposed peer collaboration activities where students collaborating synchronously or asynchronously can learn together with the teacher's lecture.

Tegos and Demetriadis [49] and Ferschke et al. [20] both proposed a similar approach by peer collaboration. Tegos and Demetriadis [49] created a system allowing teachers or researchers to register participants and assign them to their respective groups. Ferschke et al. [20], allowed a more autonomous group assignment, requiring only the approval of both peers to start the activity. Thus, the goal is to encourage students in each group to discuss a topic to submit a joint answer to an open learning question [49, 20].

Neto and Fernandes [44] proposed an online collaborative environment in groups of up to four learners, with well-defined individual roles such as group topic research, presentation paper preparation, presentation and mediation of debate, presentation of results achieved and coordination of group activities.

Guo et al. [23] proposed a collaborative programming editing in an online environment embedded with text chat system for one-on-one and small-group interactions.

Kim et al. [33] proposed small group collaboration of four to five members and medium groups of ten members. Four group discussion tasks were constructed so that each group was involved in the discussion of its task and presented a final answer extracted from that discussion. (i) Estimation task: simple estimation problems that involved inferring the height of the Eiffel Tower and the calories in an avocado without searching the Internet; (ii) Decision-making task: a travel task, widely used in group decision-making tasks. Specifically, plan a day trip to Korea for a foreign friend who was visiting the country for the first time; (iii) Open debate task: members were asked to give their opinion on a dilemma of the moral machine, in order to elicit different opinions from members. Open debates require the assessment and reasoning of ethical and social issues; and (iv) Troubleshooting task: for the participants they determined the best way to find out the name of a person you forgot without directly requesting this information (scenario involving a social context).

Göschlberger and Brandstetter [22] proposed collaborative work through chatbot-user interaction. Thus, other users have access to information through the chatbot and not by direct contact.

(RQ3) What are the advantages of collaborative activities?

Online collaborative learning offers benefits over traditional classroom learning by providing different approaches to learners and teacher interactions, enabling real-time collaboration across available learning technologies [41], offering tailored opportunities and flexibility of learning through feedback and interactive collaboration [40]. In addition to being an environment where students can express their thoughts autonomously [18].

Collaboration is identified as an important factor in successful learning in traditional and online environments [9, 41]. There are studies indicating that students perceiving collaboration in the group were more satisfied with the online learning environment [16, 18]. Also, a key factor in sustaining the educational experience is developed due to students' sense of community in developing collaboration [21].

(RQ4) How the proposal was validated?

Guo et al. [23] reviewed nine months of server logs studied, deployed as an feature in Online Python Tutor.

Allaymoun and Trausan-Matu [2] propose a evaluation based mainly on Mikhail Barkhtin's dialogism theory and Stefan Trausan-Matu's polyphonic model [51], with allow analysing the learner level of participation and Rhetorical Structure Theory (RST) trying to find out linking relations between the threads and throughout relations resulted from collaborative learning. Adding NLP techniques (tokenisation, lemmatisation and stop-words removal) and Stanford NLP tools to extract the most repeated words, allow the system to discover important threads discussed in chat.

Kim et al. [33] proposed a qualitative study with small groups, followed by a study with users with medium groups. The qualitative study involved six small groups, each consisting of four to five members. Then, a preliminary user study was carried out with two medium groups with ten members each.

At the end of each task, the participants answered a survey consisting of eight questions (about the usefulness and effectiveness of communication). After completing all tasks, participants answered three open-ended questions about the group chat experience. Also, the group behaviour (number of messages, diversity of opinions and participation), user attitudes (efficiency, effectiveness, openness of communication and usefulness) and quality of the output were evaluated.

(RQ5) What are the challenges and limitations of the proposed solution?

According to Tegos and Demetriadis [49], interpreting the results achieved should be considered a limitation of the study. In addition, learners aware that their discussions are being monitored may eventually change their typical conversational behaviour by paying more attention to the agent than they would in an uncontrolled environment. Lastly, the study results need to be confirmed by a larger sample size and other student populations of different backgrounds or ages.

Guo et al. [23] carried out nine months of studies with server logs, due to the lack of conducting a controlled experiment to formally assess the usability of the system.

Kim et al. [33] encountered several difficulties in conducting goal-oriented discussions through group chat. Reaching consensus can be more difficult in a virtual chat than in a face-to-face meeting. Controlling the participation of group members to avoid weakening the positive dynamics of the group and reducing the satisfaction of those who actively participate. Difficulties in organising different opinions because of fragmented/summarised messages, thus reducing the efficiency of the discussion.

(RQ) Summarized results:

Table 3.2 present the summarised result of the RQ1. It was evaluated in the works that although some works reported the use of management systems, chatbot was only used as a way to search and deliver content for a given request, thus making the management occur directly by the group members. It was also realised the flexibility that creating a chatbot can offer to work instead of using a chatbot tool.

Table 3.2 – Results summary RQ1

	(RQ1) What technologies are used?
[29]	Browser-based client interface approach with Express and SocketIO framework, Node.js, Jade and AngularJS
[49]	Levenshtein-based string similarity algorithm and a WordNet [42] lexicon
[15]	Virtual assistant in conjunction with the camera to capture emotions to assist students in detecting negative emotions and major problems to solve problems
[44]	Interaction between chatbot and Learning Management System (LMS)
[23]	Prototype system with WebSockets, Node.js, modified Mozilla's TogheterJS library and integrated it with the Online Python Tutor
[2]	A model for analysing (CSCL) chat collaboration using rhetorical schemes with NLP tools
[45]	Chatbot, Slack, Python, MySQL and Google Cloud Platform
[33]	Chatbot that runs in Telegram, built with BotFather. The back-end server was created with Python, using the Telegram library and pickleDB. The front-end and back-end use a Telegram dispatcher to communicate, transmit data and access APIs
[22]	A chat interface with Google's Dialogflow. The system architecture consists of several services that are communicating through different REST endpoints. Was used Postgres for database, the Elasticsearch search engine and a fulfillment

Table 3.3 present the summarised result of the RQ2. It was evaluated in the works that the number of members of a group generally depended on the type of task that the members would solve, for example, discussion groups usually have a greater number of members than groups for solving practical tasks. It was also noticed that the use of playful tasks needs more attention in management because it is difficult to reach a joint response (consensus) in a group.

Table 3.3 – Results summary RQ2

	(RQ2) Which learning activities are used: individual, collaborative (how many people), which formats (practical or playful)?
[29]	Peer collaboration activities where students collaborating synchronously or asynchronously can learn together with the teacher's lecture
[49, 20]	Peer collaboration. Each group to discuss a topic to submit a joint answer to an open learning question
[44]	Groups of up to four learners. Group for discussion and presentation of a research topic
[23]	Collaborative programming editing in an online environment embedded with text chat system for one-on-one and small-group interactions
[33]	Proposed small group collaboration of four to five members and medium groups of ten members. Group discussion tasks
[22]	Proposed collaborative work through chatbot-user interaction. Thus, other users have access to information through the chatbot and not by direct contact

Table 3.4 present the summarised result of the RQ3. It was evaluated in the works that collaborative work is strongly encouraged because it can help both in the educational development of a student and for individual growth as a person, bringing characteristics of living together in society. Learning flexibility is also seen as a positive point to be taken into account, as it is argued that people have different learning times and ways of assimilating knowledge.

Table 3.4 – Results summary RQ3

	(RQ3) What are the advantages of collaborative activities?
[41]	Different approaches to learners and teacher interactions, enabling real-time collaboration across available learning technologies
[40]	Offering tailored opportunities and flexibility of learning through feedback and interactive collaboration
[16, 18]	Students perceiving collaboration in the group were more satisfied with the online learning environment
[18]	An environment where students can express their thoughts autonomously
[21]	Sense of community is developed in collaboration

Table 3.5 present the summarised result of the RQ4. It was noticed the difficulty of evaluating proposals for the management of collaborative work. There are also different evaluations in the studies, some authors preferred to evaluate the content (conversation/discussions) of the group members and others evaluated the approach itself. Because of this difficulty, most studies interpreted the results through questionnaires.

Table 3.5 – Results summary RQ4

	(RQ4) How the proposal was validated?
[23]	Reviewed nine months of server logs studied
[2]	A evaluation based mainly on Mikhail Barkhtin's dialogism theory, Stefan Trausan-Matu's polyphonic model and NLP tools
[33]	Qualitative study with small groups, followed by a study with users with medium groups

Table 3.6 present the summarised result of the RQ5. As most of the works did not have a management system, it was realised that managing the actions of the members, especially in recreational activities, was difficult. Thus, conducting experiments and consequently interpreting the results achieved is considered a limitation of proposals for evaluating approaches in collaborative systems.

Table 3.6 – Results summary RQ5

	(RQ5) What are the challenges and limitations of the proposed solution?
[49]	Interpreting the results achieved should be considered a limitation of the study
[23]	Difficulty in carrying out a controlled experiment to formally assess the usability of the system
[33]	Reaching consensus, controlling the participation of group members and difficulties in organising different opinions

Table 3.7 shows the summary result of the research questions of our approach. Unlike other approaches in the literature where chatbot was used only as a way to retrieve information from a database or to encourage participation in the development of a project, our approach to a chatbot communicating with a MAS presents an improved management and knowledge about the proposed organisation in a collaborative work. That is, the chatbot is responsible for the communication between system and user and MAS is responsible for storing all knowledge about the organisation's management, respecting groups, roles and tasks. Therefore, although no role restrictions were used in this work, it was realised that designing a multi-agent system was a good decision due to the possibilities of organising and structuring a group, in addition to the possibility of maintaining multiple groups with their own organisation and rules. That said, there may be some difficulty in using these systems in

collaborative educational environments due to the construction of the organisation, in which roles and tasks must be well defined before the system is implemented.

Table 3.7 – Results summary our approach

	Our Approach
RQ1	Browser-based client interface approach with an integrated chatbot and a multi-agent system
RQ2	Collaborative approach to groups with eighteen members and a lecturer to solve group tasks
RQ3	Promotes greater search for knowledge
RQ4	Checking logs on the chatbot platform and with system log
RQ5	Difficulty in interpreting the results and changing the structure of a group at run-time

4. OUR APPROACH

Our approach explores the use of a chatbot in a collaborative environment, in which MAS agents represent their users as part of a project group, assisting in the organisation and communication between human users. The communication of the group members is done through the chatbot and not directly, that is, the organisation's information is available to the group members at any time. The main objective of the multi-agent system is to facilitate the dissemination of information about the current status of the project, to allow several groups to function concurrently, by allowing: greater control of restrictions and greater management by the group responsible when monitoring performance during the execution of the task without hinder the construction of knowledge.

The approach is represented in Figure 4.1. The user sends a message in natural language to the chatbot inserted on a website, when the chatbot receives a request, it is sent to JaCaMo and perceived as a plan to be executed. This plan triggered by the chatbot request checks the query parameters to redirect to a specific plan of the request. That is, when the user makes a request, several plans are triggered to return the response to the chatbot, which in turn returns the response in natural language to the user on the website.

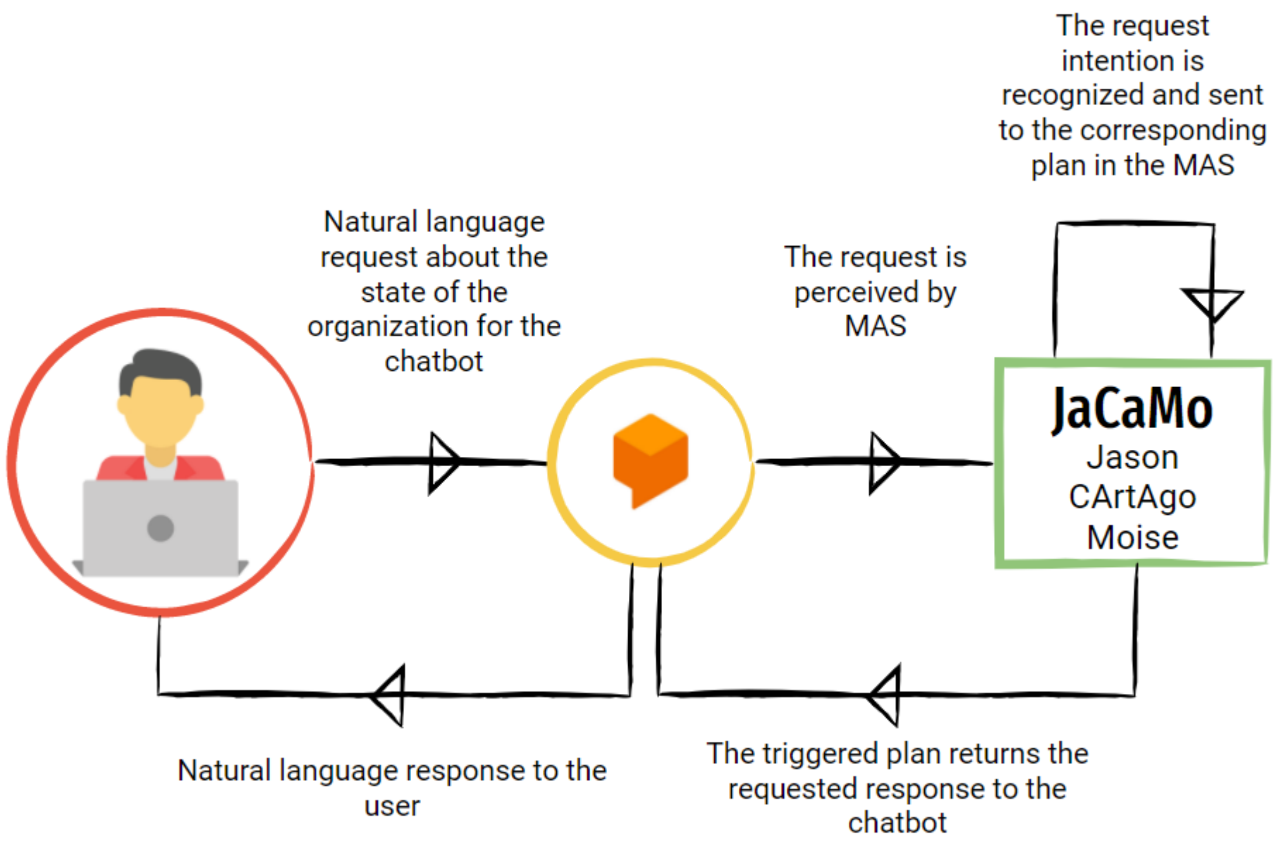

The request intention is recognized and sent to the corresponding plan in the MAS
Natural language request about the state of the organization for the chatbot
The request is perceived by MAS
JaCaMo
Jason
CArtAgo
Moise
Natural language response to the user
The triggered plan returns the requested response to the chatbot

Figure 4.1 – Our approach

The choice of the JaCaMo platform for the approach with multi-agent systems was made mainly because it is possible to create and control an organisation through Moise, allowing us to manage an organisation in terms of members and in the management of task resolution and communication with the user through a chatbot it allows a more effective and natural communication with the system.

The coordinating Jason agent has plans to receive and send requests for the chatbot and plans for all chatbot intentions regarding the group's status, which in turn redirects to the plans regarding an agent with specific group plans. If specific plans are successfully executed, the answer is returned to the coordinator, who in turn returns to the chatbot.

The system is validated by the experience of users of the software engineering discipline to test the functioning and capacity of the system. The system will assist students in a project developed in the discipline, two groups are assisted simultaneously, these groups are composed of eighteen students and a responsible teacher. Each group has different tasks to be solved and different final objectives.

4.1 Architecture

The overall architecture of the proposed system is shown in Figure 4.2. It involves software development in different environments to allow an improved user interaction and coordination. Our system is able to receive requests in the form of questions (about the state of the organisation) or actions to be performed (register new tasks, select new tasks, among others) and return a coherent answer with the user's group through plans in the JaCaMo system.

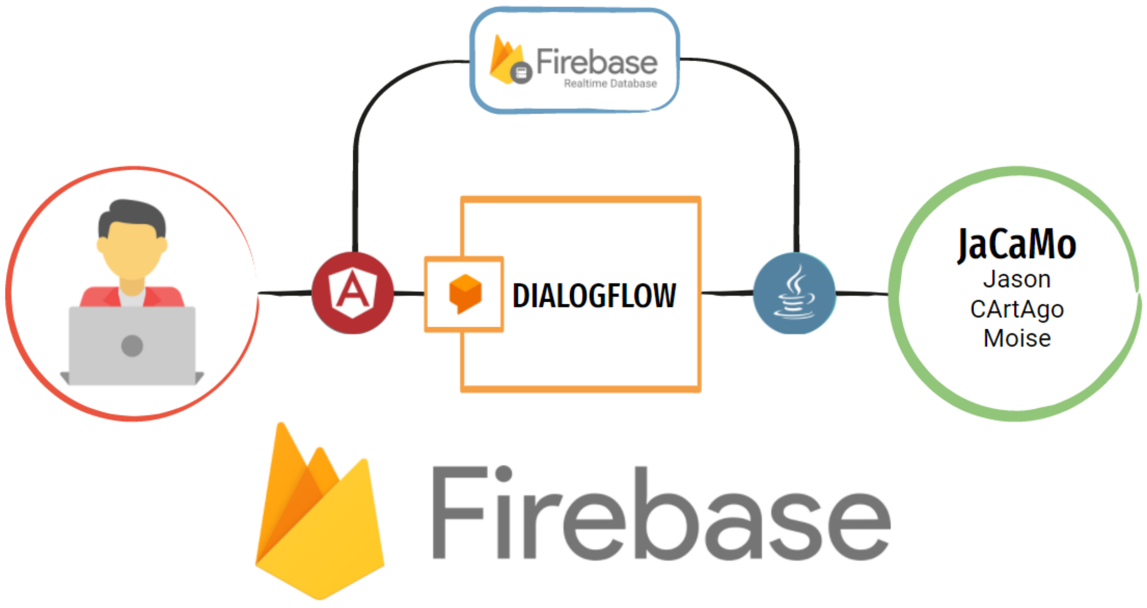


Figure 4.2 – System architecture

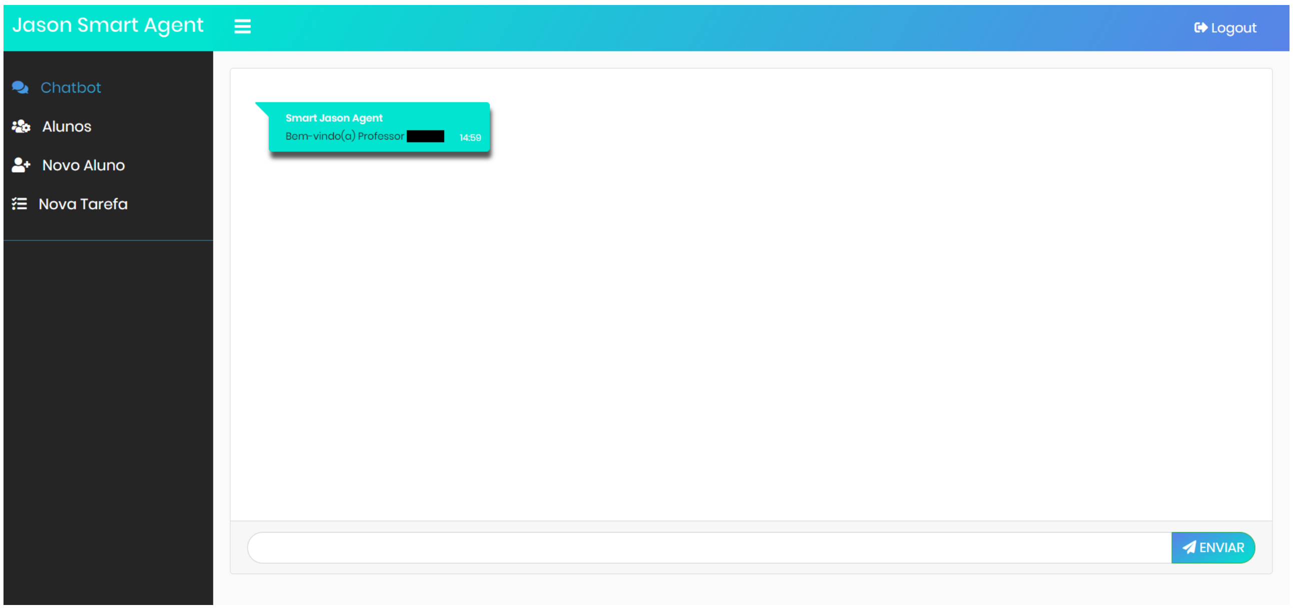
Figure 4.2 shows users' communication with the chatbot through a website and the perception of the multi-agent system with chatbot requests through a Java REST API. A database was used to allow recording of information on the user's side and to retrieve this information on the multi-agent system side.

A website was developed to define permission levels and assist in the identification of agents on JaCaMo, in addition to allowing the leader to register/delete and make requests to the chatbot. The website was developed using the Angular front-end framework, for the back-end Firebase was used with the Realtime Database.

To coordinate the groups, the multi-agent system is able to collect information in the database to initialise the groups according to commands from the leaders. The information generated at run-time by the organisation is not stored in the database, but in the artifact of each group. For the development of the chatbot, the Dialogflow platform was used. The next sections present the details of the architectures mentioned above.

4.2 Environment

A website using the Angular framework and Firebase [1] were used to facilitate the development of web applications and host the necessary information for the user. A user has two possible interfaces – depending on their level of access – with the system through the website: an interface for the administrator and leaders with forms and a chatbot, as shown in Figure 4.3, and one for students containing only the chatbot.



Smart Jason Agent
Bem-vindo(a) Professor [redacted] 14:59

Figure 4.3 – Website

[1] Available in: https://firebase.google.com/

A login page was developed to allow the JaCaMo system to correctly send the message to its respective agent, that is, the moment the user sends a message to the chatbot, information such as the user's name and role are also automatically transmitted. As a result, it is not necessary to request this information whenever the user sends a message. Also, form guides have been implemented to allow the administrator to register new students and assignments, and control guide to allow the administrator to view registered students.

To automatically send the information mentioned above, it was necessary to create a chat, instead of using the standard Dialogflow chat integration, so that the user does not see a query like *"name_last-name_role What is my group?"* but *"What is my group?"*, while in JaCaMo the complete request is sent (*name_last-name_role Question*) to extract the necessary parameters for queries. An example of how this information is processed is shown in Section 4.4.

4.3 Chatbot

The Dialogflow chatbot was used because it allowed communication between the multi-agent system and the chatbot through the conversational fulfillment (deployed as a web-hook) as shown in Figure 4.4, that calls a REST API or back-end service.

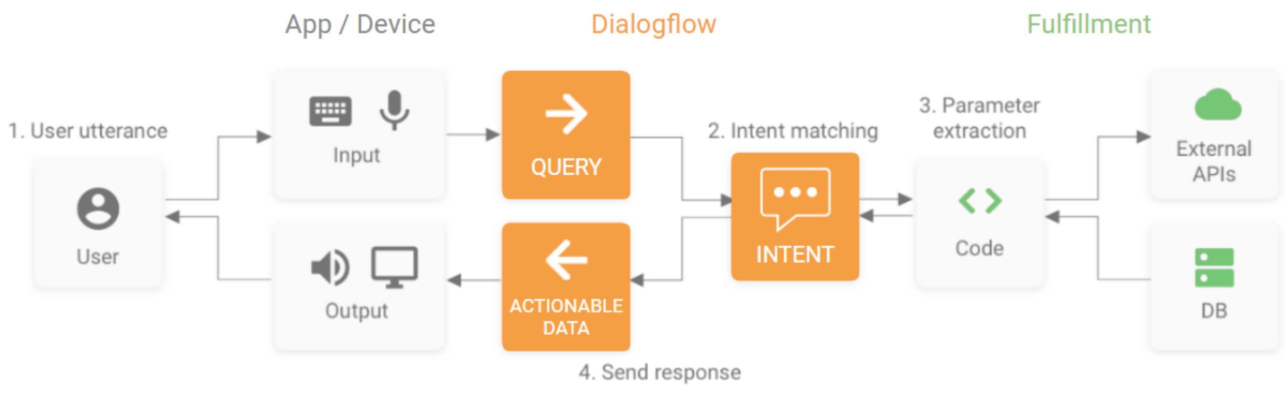
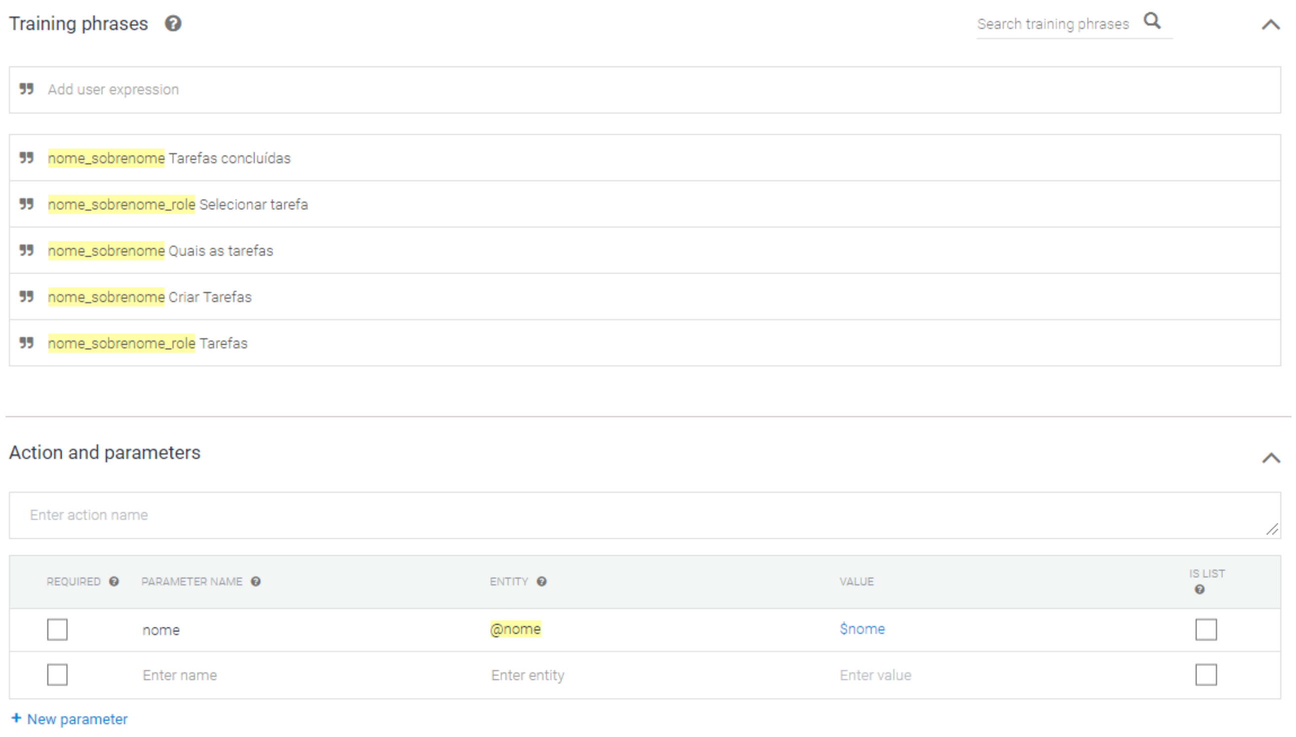


Figure 4.4 – Dialogflow architecture

The chatbot knowledge base is currently comprised of 25 intents and follow up intents (intents that continue the conversation), not counting the defaults, capable of answering questions such as: what is the group's goal; what is the role of each member of the group; which members of the group; what was the last task to be completed; when was the last update made; what are my tasks; what is the project delivery date; and how long until the task delivery date. The chatbot can also recognise requests to trigger actions on the system, such as task completion requests or multi-agent system startup. The twenty-five intents are described: commands, create group, create task, delivery date, development, difference in the delivery date, statistic, group, record, my tasks, objective, productivity, project, completed project, remove agent, response commands, groups of response commands, answer commands my tasks, response command tasks, select the group, select the log group, select the productivity group, select the task, tasks and last update.

To communicate with the JaCaMo multi-agent system, an intent was created in Dialogflow for each question to be answered and for each action to be taken. To organize requests made by users, intentions were trained using parameters. Initially four parameters were created, with the expectation of a query of the type: "*name last-name discipline role Question*": name (`@sys.given-name`), last-name (`@sys.last-name`) , discipline (`@disciplina`) and role (`@role`), but Dialogflow was unable to get the parameters with certainty, causing the multi-agent system to fail to identify which agent and plan it should trigger. With that, the phrases were trained with a parameter: name (`@name`), for queries of the type "*name_last-name_role Question*". The `@name` parameter is identified by an entity created that recognises the pattern of a word through Regex.

Figure 4.5 shows an example of task-related intent training. This example shows how the chatbot recognises questions related to tasks in order to return options of actions that the user wants to perform.



nome_sobrenome Tarefas concluídas
nome_sobrenome_role Selecionar tarefa
nome_sobrenome Quais as tarefas
nome_sobrenome Criar Tarefas
nome_sobrenome_role Tarefas

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	nome	@nome	$nome	☐
☐	Enter name	Enter entity	Enter value	☐

Figure 4.5 – Dialogflow intent training

4.4 Multi-Agent System

The system was implemented based on the JaCaMo and Dialogflow integration project developed by Engelmann et al [2]. The integration project was adapted according to the needs of the proposal project. Figure 4.6 shows how the integration with the multi-agent system is currently implemented.

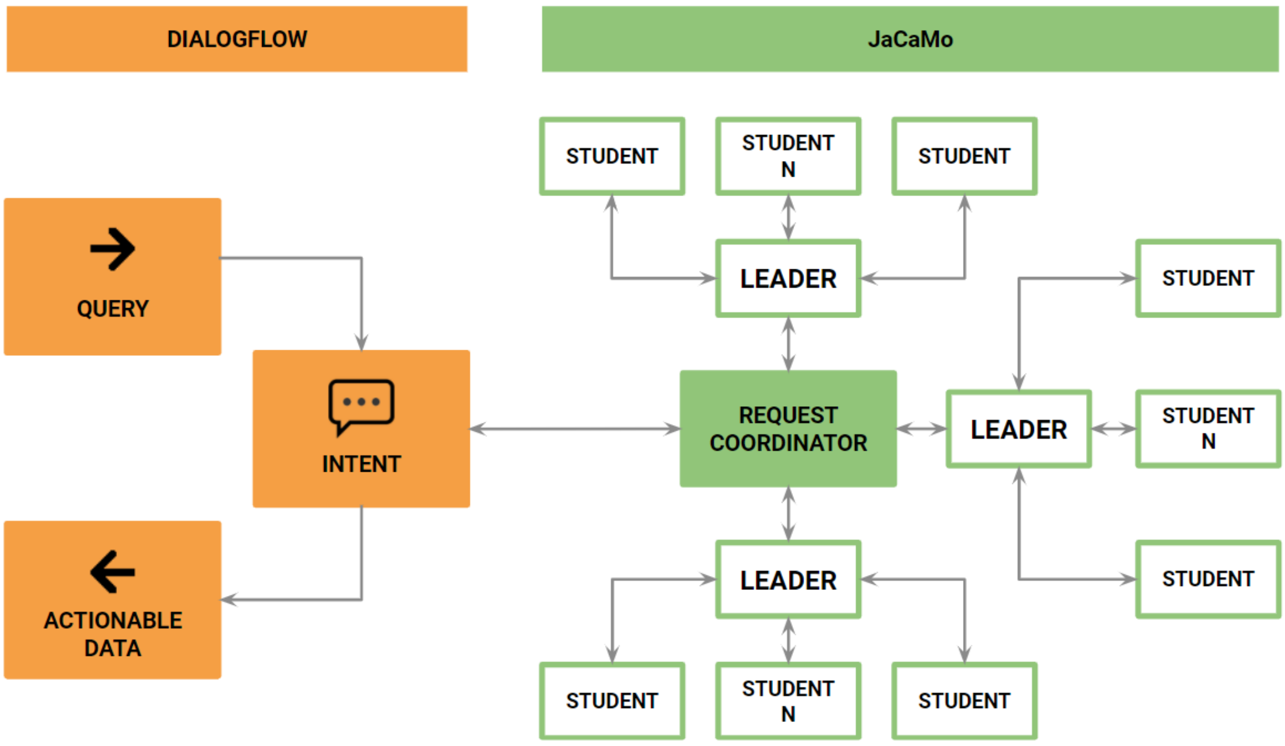


Figure 4.6 – Dialogflow and JaCaMo integration

The system was developed so that the agents of a group are responsible for a function, each with a set of different missions that must be fulfilled for the project to be completed. These roles and missions must be part of the Moise scheme, previously organised before the system started, therefore, they must be modified according to the instructions and needs of the responsible teacher. Moise is responsible for the student to know that he can already perform his task.

The multi-agent system is responsible for storing all the information of the group, which tasks each student has to solve, which is the student's role in the group, the objectives and members of the group. The chatbot is the means for students to extract information from the multi-agent system in natural language. The multi-agent system receives a POST from the REST API that treats the information that comes from the chatbot as an object. This information is refactored to extract the necessary parameters and initiate an internal action

[2]https://github.com/DeboraEngelmann/helloworld_from_jason

in Jason, which consequently the system of continuity in the communication between agents to meet the user's request.

For the development of the JaCaMo multi-agent system, three types of agents were used: **REQUEST COORDINATOR** agent responsible for the requests that the system receives from Dialogflow and for managing information with Firebase; **LEADER** agent responsible for group coordination, that is, it is the agent that does all the work related to the group's initialisation (creation of the group artifacts and workspace), initialises the agents of your group and deals with the management of the requests of the group; and **STUDENT** is the agent responsible for solving tasks, with this, it handles its own requests regarding tasks and redirects requests regarding the group to its leader.

The system works by communicating between agents and by reaching plans according to the perception of requests. The request coordinator controls the actions between agents through Jason's communications, in which these communications have two formats, which can be just a tell to send beliefs to another agent or an achieve, telling another agent to execute a certain plan. Chatbot requests are perceived by the request coordinator through internal operations in the CArtAgo artifact. Each perceived request executes the `request` plan, which forwards it to the `response` plan. There is a response plan for each chatbot intent, which executes different plans, based on the parameters received from the triggered intent.

For example, when the request coordinator receives an intention to create a group, it triggers the `setup` plan that uses a CArtAgo operation to get the group information in Firebase. When the artifact takes the information, an internal operation is triggered to create a leading agent, send beliefs to that agent and communicated to that agent to execute a setup plan. In turn, the lead agent creates the workspace, the group organisation, the group scheme and executes a plan for creating student-type agents. As the students are created in the system, the lead agent sends orders to execute plans for configuring the workspace, organisation and schema. Finally, to complete the group creation, the lead agent stores the information of the tasks present in Moise.

Regarding tasks requests, the request coordinator communicates directly to the agent responsible for the request. The plans responsible for the tasks are defined to respect the Moise organisation, so there is always a permission check for the student's actions. That is, the system checks whether the agent is able to perform the proposed action, checking the status of the task, the role that the agent is representing in the group and the very activities that the agent is performing or not.

And finally regarding the group's information requests, the group leader keeps the organisation's consultation information in its belief base. With that, all requests for group information that are directed to student agents are redirected to the leader.

Figure 4.7 shows how the architecture works through an example of user-system interaction. The student asks a question regarding the tasks he is responsible for performing

(*"What are my tasks?"*). The chatbot recognises the user's intention (*My Tasks*) and extracts parameters such as the student's full name (*name: John, last-name: Doe*) and if he is a project leader and forwards the query to the multi-agent system. The multi-agent system recognises these parameters and forwards them to the agent responsible for representing their real user, activating plans responsible for handling the request and forwards the query response to the chatbot, which in turn presents the response to the user (*"Responsible for the Task: Register users in the system"*).

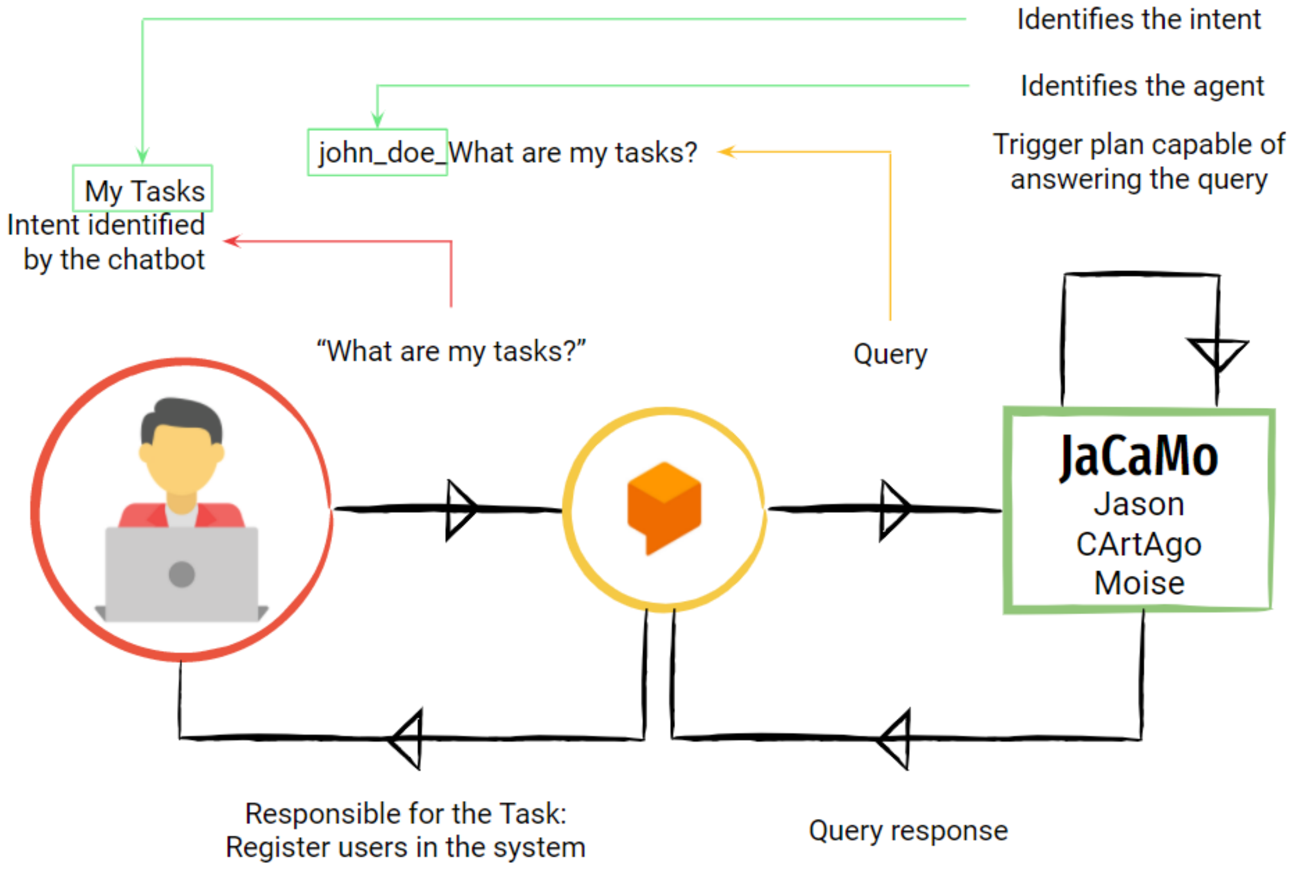

Identifies the intent
Identifies the agent
Trigger plan capable of answering the query
john_doe_What are my tasks?
My Tasks
Intent identified by the chatbot
"What are my tasks?"
Query
JaCaMo
Jason
CArtAgo
Moise
Responsible for the Task: Register users in the system
Query response

Figure 4.7 – Example interaction

Figure 4.8 shows how the system flows, the steps are described below:

1 User asks a question to the chatbot about the group's tasks (*"What tasks to solve"*);

2 Question triggers an intent. The chatbot identifies the parameters and the intent in the query;

3 Query is sent to JaCaMo via web-hook by a POST request;

4 Information is processed in Java and sent to the multi-agent system;

5 Request Coordinator verifies the parameters received from the chatbot such as: name of the agent and the intent and sends it to the agent that represents the student in the multi-agent system;

6 Agent asks the Leader of his group for information about the tasks;

7 Leader sends the information requested by the Student to the Request Coordinator;

8 Multi-agent system sends the response to the chatbot;

9 Chatbot responds to the user the task to be solved.

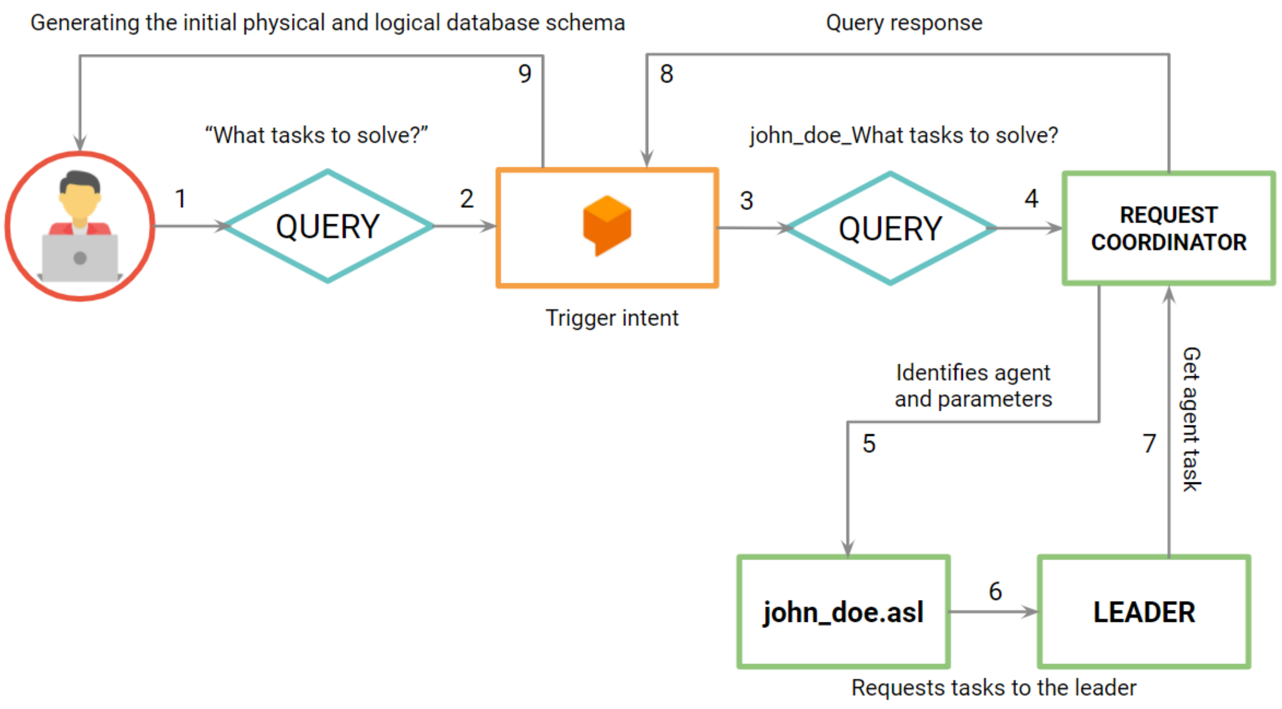

Generating the initial physical and logical database schema
Query response
9
8
"What tasks to solve?"
john_doe_What tasks to solve?
1
QUERY
2
3
QUERY
4
REQUEST COORDINATOR
Trigger intent
Identifies agent and parameters
5
7
Get agent task
john_doe.asl
6
LEADER
Requests tasks to the leader

Figure 4.8 – Interaction flow

Request coordinator is the agent who is focused on the artifact referent to the requests of Dialogflow. It has plans to deal with each Dialogflow intent, that is, for each chatbot intent there is a corresponding plan. As it receives all requests from Dialogflow, the agent is responsible for the communication between the real user and his respective agent, that is, it forwards the information to the respective agent. It also has plans responsible for starting a leading agent at run-time, sending the necessary beliefs for that agent to start his group.

Listing 4.1 shows an example of a request to the coordinator. The `+request` event correspond to chatbot requests to the multi-agent system, `+!responder` are plans responsible for triggering specific plans based on the perceived intentions of the chatbot requests. In this example, the triggered plan is to show the commands available to the user, so the `+!comandos` plan is triggered and creates a context (`contextBuilder`) to trigger a chatbot conversation sequence and then sends the response to the user via the `+!answer` plan. With the context referenced, the chatbot will give options of actions to be performed in the system, making the user's response trigger specific plans related to the context.

Listing 4.1 – Request coordinator example

```
+request(ReponseID, IntentName, Params, Contexts): true <-
    !getNameByParam(Params, Name);
    systemLog(IntentName, Name);
    !responder(ResponseId, IntentName, Params, Contexts);
.

+!responder(ResponseID, IntentName, Params, Contexts):
    (IntentName == "Comandos")
    <-
    !getNameByParam(Params, Name);
    !comandos(Name);
.

+!comandos(Nome):
    .all_names(L) &
    .term2string(N, Nome) &
    .member(N, L)
    <-
    contextBuilder("respostaComandosContext", Contexto);
    !answer("0. Cancel<br>
        1. Tasks<br>
        2. My Tasks<br>
        3. Group",
        Contexto
    );
.

+!answer(Resposta, Contexto): true <-
    reply(Resposta, Contexto);
.
```

Listings 4.2, 4.4, 4.5, 4.6, 4.8 and 4.9 describe the creation and configuration of multiple groups, that is, these plans are responsible for the entire configuration of the organisation, as long as the system already has a Moise organisation previously configured.

Listing 4.2 shows an execution of a plan when perceiving a group creation request `+group`. Upon realising the creation of a group, the `+!createLeader` plan is triggered by creating the initialised project leader and sending beliefs to the leader and sending him to reach the setup plan.

Listing 4.2 – Create group request

```
+grupo(start(Leader,Grupo)): true <-
     !createLeader;
.

+!createLeader:
     grupo(start(Leader, Grupo) &
     projeto(Projeto) &
     descricao(Objetivo)
     <-
     .create_agent(Leader, "leader.asl");
     .send(Leader, tell, start(Grupo, Projeto, Objetivo));
     .send(Leader, achieve, setup);
     +grupoStatus(Leader);
.
```

Listing 4.3 shows an example of a follow up intent in the execution of a plan responsible for verifying parameters when receiving a command intent. This plan verifies the options given by the command intent and verifies if it is a student to create a new conversation context for the continuity of the operation (follow up intent).

Listing 4.3 – Example follow up intent

```
+!executar(Nome, Operacao):
     Operacao == 1 &
     .all_names(L) &
     .term2string(N, Nome) &
     .member(N, L)
     <-
     contextBuilder("respostaComandosTarefasContext,Contexto);
     !answer("0. Cancel<br>
          1. List registered tasks<br>
          2. Select task<br>
          3. List unfinished tasks",
          Contexto
     );
.

+!executar(Nome, Operacao):
     Operacao == 2 &
```

```
            .all_names(L) &
            .term2string(N, Nome) &
            .member(N, L)
            <-
            contextBuilder("respostaComandosMinhasTarefasContext,
                Contexto
            );
            !answer("0. Cancel<br>
                1. Task in progress<br>
                2. Mark task as done<br>
                3. Release task<br>
                4. Task delivery date<br>
                5. How much to deliver the task,
                Contexto
            );
        .
```

Leader is the agent who is responsible for starting the group at run-time, creating the members of your team dynamically with the information provided by the request coordinator. It has plans responsible for: initialising and defining the team's workspace, group and initial tasks; in addition to initialising team members, he is also responsible for assigning missions and roles to these agents; and finally is responsible for answering the questions regarding the group, that is, organising the group's information and forwarding it to the request coordinator according to the consultations. The commands available to the group leader are described bellow:

- `commands`: show the commands available on the system (tasks [register task, remove task, list tasks, tasks not being performed and uncompleted tasks], group [group members, list due date for tasks, remove member and mark project as complete], log and members' productivity);
- `create group`: initialise the group on MAS;
- `tasks`: show commands for tasks (register task, remove task, list tasks, tasks not being performed and uncompleted tasks);
- `group`: show commands for group (group members, list due date for tasks, remove member, mark project as complete);
- `log`: show the group log;
- `productivity`: show group productivity;

- `update`: show the last update made by the group;
- `objective`: show the project description;
- `delivery date`: list the due date for tasks;
- `completed project`: mark the project as completed.

Figure 4.9 shows how information is forwarded by the request coordinator to the group leader. The leader makes a request for consultation – in case of an administrator user it is also possible to make a request for group creation – to the corresponding leader, if necessary it seeks information from the members of the group itself and returns the information to the chatbot or if there is already information in their own beliefs, information is returned directly.

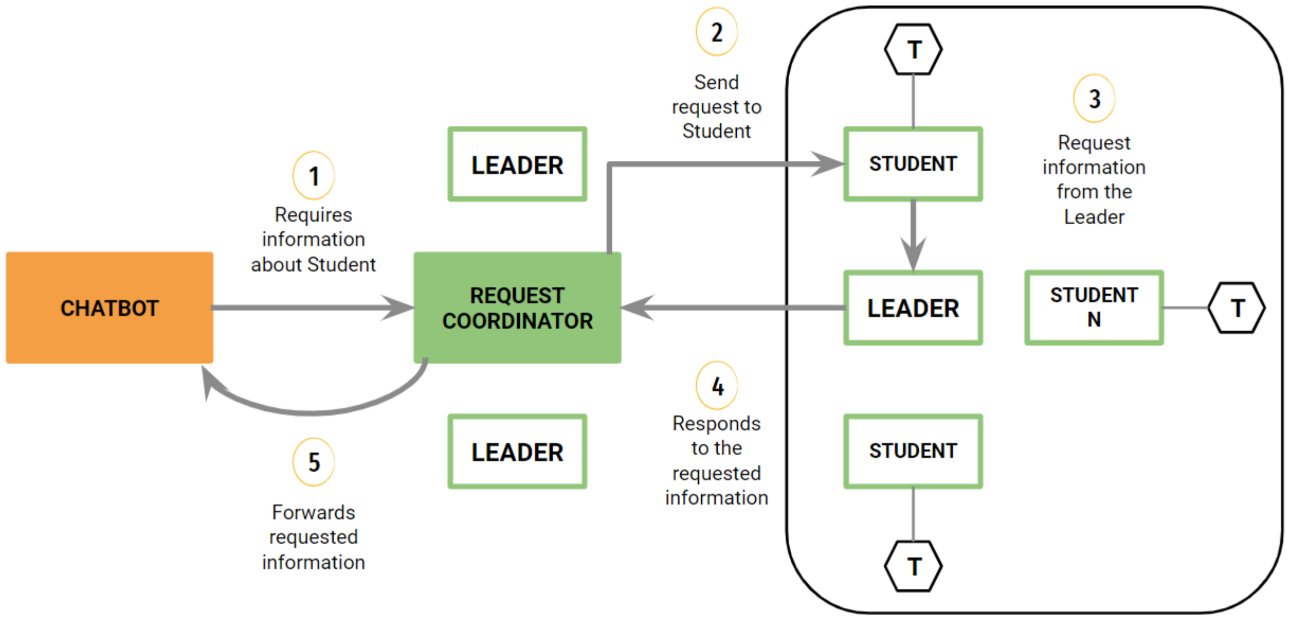

CHATBOT
1
Requires information about Student
REQUEST COORDINATOR
LEADER
LEADER
2
Send request to Student
STUDENT
3
Request information from the Leader
LEADER
STUDENT N
STUDENT
T
T
T
4
Responds to the requested information
5
Forwards requested information

Figure 4.9 – Leader requests

Listing 4.4 shows the plan responsible for starting the group and the agents. Is initialised the workspace (`createWorkspace` and `joinWorkspace`), artifact (`makeArtifact`), organisation (`createGroup`) and the plan to initialise agents is triggered (`!createAgents`). This plan ensures that each group has its own artifacts and workspace, in addition to adopting the position of leader to the agent (`adoptRole`). The workspace artifact provides functionalities to create workspace (`createWorkspace`), create group (`createGroup`), join (`joinWorkspace`), lookup (`lookupArtifact`), make artifcat (`makeArtifact`), focus artifacts of the workspace (`focus`) and also it provides operations to set roles (`adoptRole`). We use `focus` in order to perceive the artifact.

Listing 4.4 – Group setup

```
+!setup:
    .my_name(Nome) &
    start(Agentes,Projeto,_)
    <-
    .concat("workspace_", Nome, WName);
    createWorkspace(WName);
    joinWorkspace(WName, WOrg);
    .lenght(Agentes, Ag);
    .concat("tarefa_", Nome, ArtName);
    makeArtifact(ArtName, "board.Tasks", [Ag],
        OrgArtId[wid(WOrg)]
    );
    focus(OrgArtId[wid(WOrg)]);
    joinWorkspace(Nome, GrOrgId);
    .concat("time_", Nome, GroupName);
    createGroup(GroupName, team, GrArtId)[artifact_name(Nome),
        wid(GrOrgId)];
    adoptRole(leaderRole)[artifact_id(GrArtId)];
    focus(GrArtId)[wid(GrOrgId)];
    !createAgents;
.
```

Listing 4.5 shows the plan responsible for creating the agents (`.create_agent`), the plan to trigger the specific agent and initialise the artifacts previously created by the leader, the schema initialisation and also triggers the plan to assign the mission. First, all agents in the group are created and initialised by the setup plan. When all agents are fully initialised, the group schema is created, which is a global objective decomposition tree (`createScheme`), add a scheme under the responsibility of a group (`addScheme`) and the process of assigning missions is started (`!assignMissions`). `lookupArtifact` is used to search the group's artifacts and `?formationStatus` is used to check if the group is well formed.

Listing 4.5 – Create agents

```
+!createAgents:
    .my_name(Nome) &
    start(Agentes,_,_) &
    qtdAgentes(QtdAgentes) & QtdAgentes > 0
    <-
    .nth(QtdAgentes-1, Agentes, Agente);
```

```
        .create_agent(Agente, "student.asl");
        .concat("tarefa_", Nome, ArtName);
        .concat("time_", Nome, GroupName);
        .send(Agente, achieve, setup(ArtName, GroupName));
        descontaAgentes;
        !createAgents;
    .

    +!createAgents:
        .my_name(Nome) &
        qtdAgentes(QtdAgentes) & QtdAgentes == 0
        <-
        lookupArtifact(Nome, GrOrgId);
        .concat("esquema_", Nome, Scheme);
        createScheme(Scheme, taskScheme, SchArtId)[wid(GrOrgId)];
        focus(SchArtId)[wid(GrOrgId)];
        .concat("time_", Nome, GroupName);
        lookupArtifact(GroupName, GrArtId);
        ?formationStatus(ok)[artifact_id(GrArtId)];
        addScheme(Scheme)[artifact_id(GrArtId)];
        resetaAgentes;
        !assignMissions;
    .
```

Listing 4.6 shows the plan responsible for triggering the specific agent and initialising the artifacts previously created by the leader and is also responsible for saving all the group's goals. First we use `send` for all agents in the group to achieve the `setMission` plan. After all agents are correctly configured, a list is generated with all the objectives of the group.

Listing 4.6 – Assign agent mission

```
    +!assignMissions:
        .my_name(Nome) &
        start(Agentes,_,_) &
        qtdAgentes(QtdAgentes) &
        QtdAgentes > 0
        <-
        .nth(QtdAgentes-1, Agentes, Agente);
        .concat("esquema_", Nome, Scheme);
```

```
          .send(Agente, achieve, setMission(Scheme));
          descontaAgentes;
          !assignMissions;
     .

     +!assignMissions:
          qtdAgentes(QtdAgentes) & QtdAgentes == 0
          <-
          .findall(Goal, specification(
               scheme_specification, taskScheme,
               goal(_,_,_,_,_,_,plan(_,Goal)),
               _,
               _
          )), Goals);
          .term2string(Goals, Objetivos);
          missoesMoise(Objetivos);
          resetaAgentes;
     .
```

Listing 4.7 shows the plan responsible for responding to a request for the last change (date and task performed) in the group and sends it to the coordinator to trigger the plan to respond to the chatbot.

Listing 4.7 – Example leader request

```
     +!getLastUpdate: true <-
          ?ultimoUpdate(Data);
          ?ultimaTarefaCompleta(Tarefa);
          .concat("[", Data, "]:  ", Tarefa, Resposta);
          .send(chatbot, achieve, answer(Resposta));
     .
```

Student is the agent who has plans to dynamically join a group and workspace as directed by the group leader. It also has plans related to the tasks to be completed for the resolution of the group project, that is, the plans are related to the situation of the tasks in relation to the Moise scheme, such as: selection, drop and completion of tasks. And finally, it is also able to directly answer to request coordinator some queries regarding tasks. The commands available to students are described bellow:

- `commands`: show the commands available on the system (tasks [list registered tasks, select task and list unfinished tasks], my tasks and group [group members and project status (completed or not completed)]);

- `tasks`: show commands for tasks (list registered tasks, select task and list unfinished tasks);
- `group`: show group members;
- `update`: show the last update made by the group;
- `objective`: show the project description;
- `delivery date`: list the due date for tasks;
- `how much to deliver the task`: show how much is left for delivery date;
- `project`: show the state of completion of the project;
- `concluded`: mark task as complete.

Figure 4.10 shows in relation to a student request. If the request is for consultation, the Jason agent corresponding to the student who made the request sends a message to the group leader to respond to that request. If the request is to update the group status, the operation is performed only at the student's Jason agent.

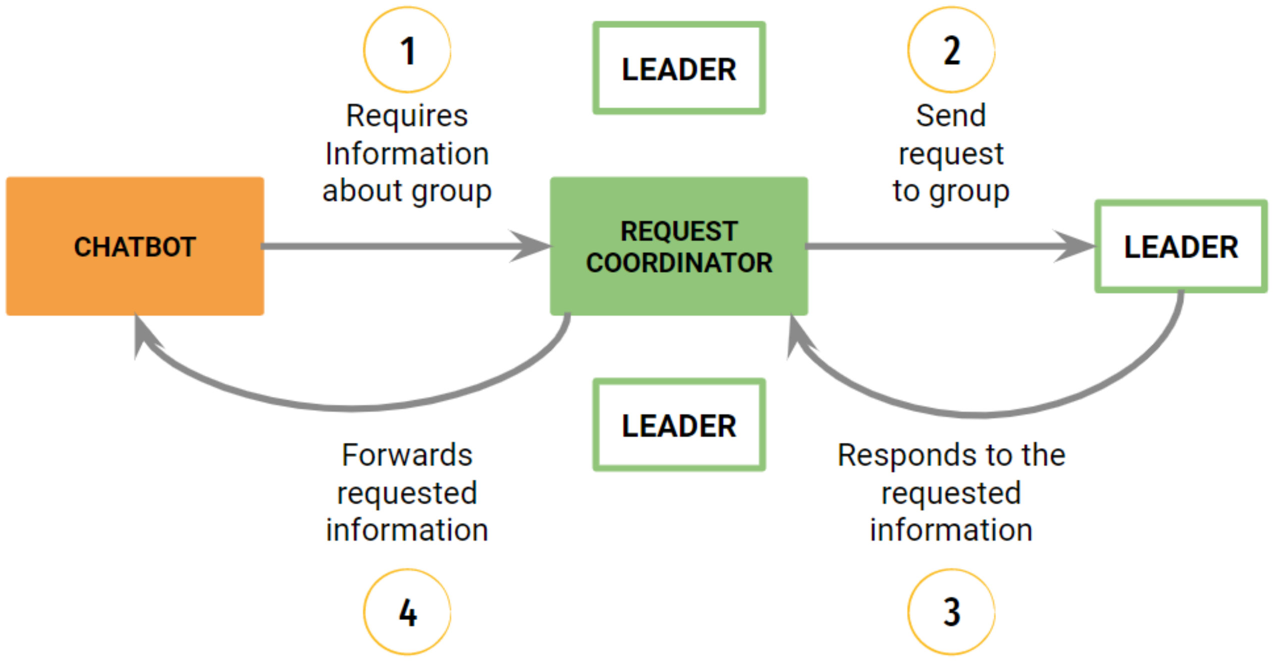

1
Requires Information about group
LEADER
2
Send request to group
CHATBOT
REQUEST COORDINATOR
LEADER
LEADER
Forwards requested information
Responds to the requested information
4
3

Figure 4.10 – Student requests

The tasks that students had to solve were related to software development, that is, front-end, back-end development tasks, among others. The following are examples of tasks to be performed: modelling the persistence layer, generating the initial physical and logical database schema, creating the POST method to receive the registration data, creating the

GET endpoint in the users API, creating the profile screen and user route management. The complete list of tasks can be found in the Appendix A.

Listing 4.8 shows the plan responsible for initialising the agent artifacts. When the group leader sends a message to the agent to reach the `setup` plan, artifacts related to the group are searched for (`lookupArtifact`), focused (`focus`) and the agent's role is also selected (`adoptRole`).

Listing 4.8 – Agent setup

```
+!setup(OrgArt, GrArt)[source(Nome)] <-
    .concat("workspace_", Nome, WName);
    joinWorkspace(WName, WOrg);
    lookupArtifact(OrgArt, OrgArtId);
    focus(OrgArtId)[wid(WOrg)];
    joinWorkspace(Nome, GrOrgId);
    lookupArtifact(GrArt, GrArtId);
    focus(GrArtId)[wid(GrOrgId)];
    adoptRole(studentRole)[artifact_id(GrArtId)];
.
```

Listing 4.9 shows the plan responsible for initialising the agent's artifacts for the mission. First the group's artifacts are recognised (`lookupArtifact`) and then the focus is made (`focus`). This plan is triggered by the group leader.

Listing 4.9 – Set agent mission

```
+!setMission(SchArt)[source(Nome)] <-
    lookupArtifact(SchArt, SchArtId);
    lookupArtifact(Nome, GrOrgId);
    focus(SchArtId)[wid(GrOrgId)];
.
```

Listing 4.10 shows the plan responsible for committing to carry out the proposed mission. When this plan is triggered first it checks if the task that the agent selected is a main task (task that is in the Moise) or if it is a secondary task (inserted later by a group leader) by the operation (`missoes`) and then it is triggered the plan `!assignTask`. The `!assignTask` plan causes the agent to commit to a mission in the scheme (`commitMission`) and send a completion response to the user.

Listing 4.10 – Agent assign task

```
+!assignTask(DataAtual, Hora, Agente, Tarefa): true <-
     .findall(Mission, specification(
          scheme_specification(taskScheme,_,Mission,_)
     ), Missions;
     .term2string(Missions, Missoes);
     .concat("mission_tarefa_", Tarefa, Missao);
     missoes(Missao, Missoes, IsMission);
     !atribuiTarefa(DataAtual, Hora, Agente, Tarefa, IsMission);
.

+!atribuiTarefa(DataAtual, Hora, Agente, Tarefa, IsMission):
     play(Leader,eaderRole,_) &
     IsMission
     <-
     .concat("mission_tarefa_", Tarefa, Mission);
     .term2string(Missao, Mission);
     .concat("esquema_", Leader, Scheme);
     commitMission(Missao)[artifact_name(Scheme)];
     atribuirTarefa(DataAtual, Hora, Agente, Tarefa, Resposta);
     .send(chatbot, achieve, answer(Resposta));
.
```

Listing 4.11 shows the plan responsible for releasing the task, and it is no longer necessary to perform it. First, the plan verifies whether the corresponding agent is really committed to carrying out the mission (`commitment`), if the agent is committed, `leaveMission` is used to remove the mission from the scheme and then a confirmation response is sent to the user.

Listing 4.11 – Agent drop task

```
+!dropTask(DataAtual, Hora, Nome):
     .term2string(Ag, Nome) &
     commitment(Ag, Mission, Scheme)
     <-
     leaveMission(Mission)[artifact_name(Scheme)];
     liberaTarefa(DataAtual, Hora, Nome, Resposta);
     .send(chatbot, achieve, answer(Resposta));
.
```

Listing 4.12 shows a task completion request, it is the responsible plan to achieve the objective. First, the plan checks whether it is a primary mission (`IsMission`), whether the corresponding agent is really committed to carrying out the mission (`commitment`) and the current active obligations (`obligation`), whether these checks are fulfilled, the agent sets the goal as performed by him (`goalAchieved`) and a response message is sent to the user. It is an action plan, so the communication takes place directly between the student agent and the request coordinator agent.

Listing 4.12 – Example agent request

```
+!finalizarTarefa(DataAtual, Tarefa, Resposta,
     Objetivo, IsMission):
     .my_name(Ag) &
     IsMission &
     play(Leader,leaderRole,_) &
     commitment(Ag,_,_) &
     obligation(Ag,Norm,What,Deadline)[artifact_id(ArtId)]
     <-
     .concat("[", Nome, "]:  ", Tarefa, Task);
     !attLeaderBelief(Task, DataAtual);
     .concat("tarefa_", Objetivo, Go);
     .term2string(Goal, Go);
     goalAchieved(Goal)[artifact_id(ArtId)];
     .send(chatbot, achieve, answer("Good job!"));
.
```

Listing 4.13 shows a simplified example of organisation of the Moise scheme in which there are only a few examples of tasks and there are no dependencies between them, that is, the tasks have no prerequisites to be fulfilled. Thus, each goal has a specific mission, in which multiple students can complete the task (minimum 0, there may be no one performing the task at the moment, and maximum 18, maximum number of students).

The organisation's specification is divided into: structural-specification, functional-specification and normative-specification:

- Structural-specification: defines the roles and the number of agents that can fill those roles;
- Functional-specification: defines the organisation's scheme, the objectives and the missions to be accomplished are defined;
- Normative Specification: states both the required roles for missions and missions obligations for roles.

Listing 4.13 – Moise scheme

```
<organisational-specification
  id="groupOrg" os-version="0.8"
  xmlns='http://moise.sourceforge.net/os'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation='http://moise.sourceforge.net/os
                    http://moise.sourceforge.net/xml/os.xsd'>

  <structural-specification>
    <role-definitions>
      <role id="leaderRole" />
      <role id="studentRole" />
    </role-definitions>

    <group-specification id="team">
      <roles>
        <role id="leaderRole" min="1" max="1" />
        <role id="studentRole" min="1" max="18" />
      </roles>
    </group-specification>
  <structural-specification>

  <functional-specification>
    <scheme id="taskScheme">
      <goal id="complete_Project" ds="Project#1">
        <plan operator="parallel">
          <goal id="task_1" ds="Import photos"></goal>
          <goal id="task_2" ds="Add photo description"></goal>
          <goal id="task_3" ds="US16 - Create profile screen">
          </goal>
          <goal id="task_4" ds="US16 - Create screen components">
          </goal>
          </goal>
          <goal id="task_5" ds="US16 - Integrate screen
                                  and components">
          </goal>
          <goal id="task_6" ds="US16 - Create GET endpoint
                                  to receive company data">
          </goal>
        </plan>
      </goal>

      <mission id="mission_task_1" min="0" max="18">
        <goal id="task_1" />
      </mission>

      <mission id="mission_task_2" min="0" max="18">
        <goal id="task_2" />
      </mission>

      <mission id="mission_task_3" min="0" max="18">
        <goal id="task_3" />
      </mission>

      <mission id="mission_task_4" min="0" max="18">
        <goal id="task_4" />
      </mission>

      <mission id="mission_task_5" min="0" max="18">
        <goal id="task_5" />
      </mission>

      <mission id="mission_task_6" min="0" max="18">
        <goal id="task_6" />
      </mission>
    </scheme>
  </functional-specification>

  <normative-specification>
    <properties>
      <property id="mission_permission" value="ignore"/>
      <property id="mission_left" value="ignore"/>
    </properties>
  </normative-specification>

</organisational-specification>
```

5. EXPERIMENT RESULTS AND ANALYSIS

Our system was evaluated during two months of tests with students in the discipline of software engineering. It was tested with two groups (two software development projects), in which the groups consisted of eighteen students and a lecturer responsible for the group. The students performed several sprints during the development of the project, with that, with each sprint the new tasks were updated in the multi-agent system, specifically in the Moise organisation, as well as corrections and improvements in the system's response.

The students' questions and answers were checked through the Dialogflow platform itself to assess whether the chatbot was able to correctly identify and trigger the desired intent. The number of requests, matches and non-matches of the chatbot can be checked in Table 5.1.

Table 5.1 – Chatbot requests

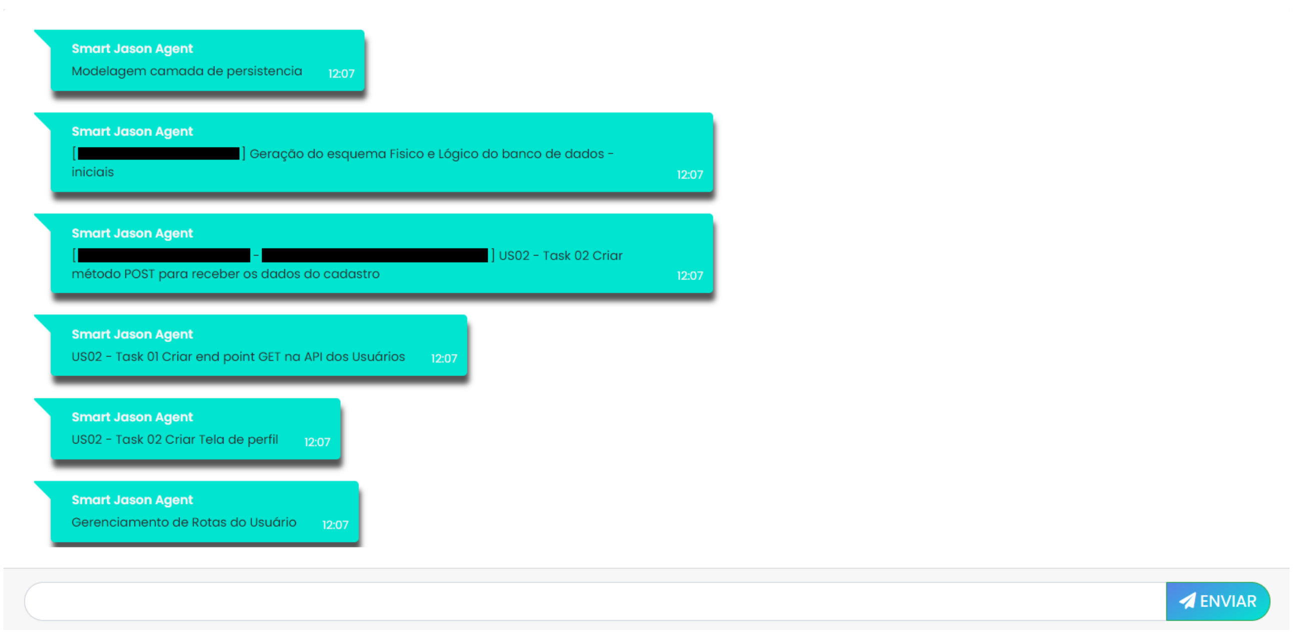
	Requests	**No Match**
September, 2020	**299**	**53**
October, 2020	**278**	**1**
Total	**577**	**54**

Due to the multi-agent system, as its name says, being able to support the management of multiple agents when communicating with the chatbot to manage individual information, the chatbot initially had difficulty in triggering a certain intention due to the way the parameters were extracted from the information of a student consultation. After identifying the problem, the solution presented in Section 4.3 was proposed to guarantee the correct extraction of the parameter that identified the student, in order to correspond with his agent in the system.

The multi-agent system was able to deliver quick responses (as a rule, responses should take less than two seconds, necessary due to the time that Dialogflow waits for a query to be answered, but responses usually take less than a second) and consistent with the group and its representation in the system. The exchange of messages between the agents of the system to consult information about the group was an interesting approach regarding the organisation, since all information about a group was stored in one place, specifically with the group leader.

Figures 5.1 and 5.2 show a request for listing the tasks available to the group (students' names were censored for privacy reasons). The answer to this request is valid both for listing tasks and for selecting a task.

The agent answers this query by returning a conversation bubble for each task in the formats: **[*Agent1* Performing *Task* - *AgentN* Performing *Task*] *Task*** (if there are agents doing the task) or just **Task** (if there are no agents doing the task).



Smart Jason Agent
Modelagem camada de persistencia 12:07

Smart Jason Agent
[[redacted]] Geração do esquema Fisico e Lógico do banco de dados - iniciais 12:07

Smart Jason Agent
[[redacted] - [redacted]] US02 - Task 02 Criar método POST para receber os dados do cadastro 12:07

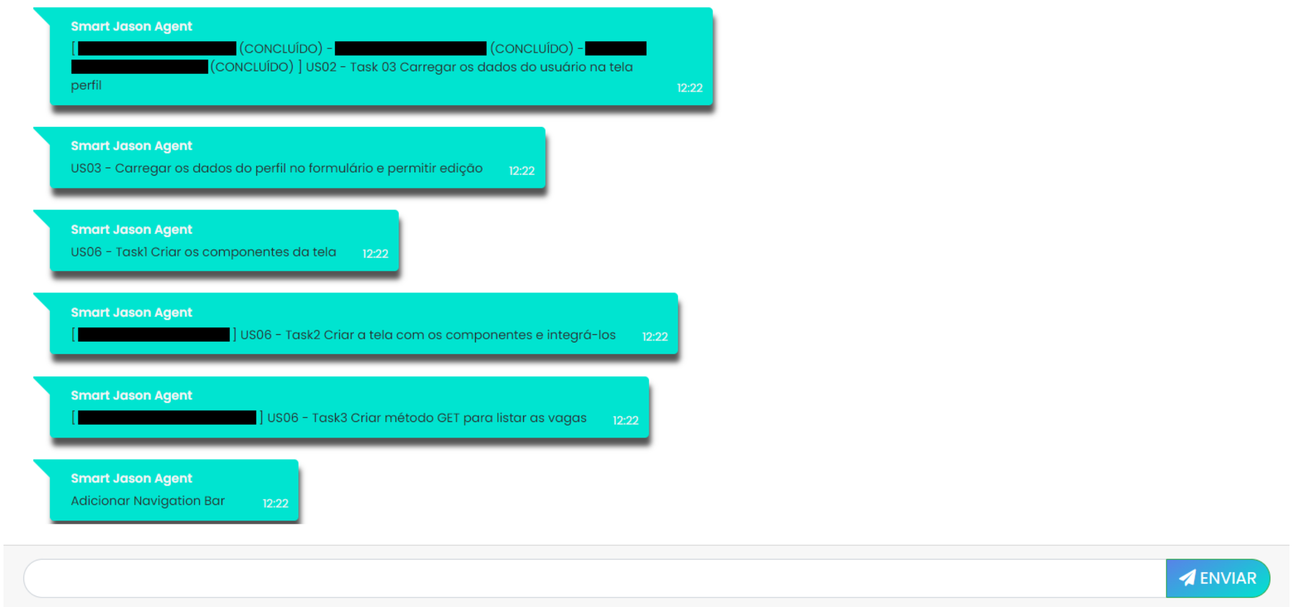
Smart Jason Agent
US02 - Task 01 Criar end point GET na API dos Usuários 12:07

Smart Jason Agent
US02 - Task 02 Criar Tela de perfil 12:07

Smart Jason Agent
Gerenciamento de Rotas do Usuário 12:07

Figure 5.1 – Task List Request

The status of completion of the task is also shown in case the student responsible for performing the task completes the task, as shown in Figure 5.2. In case of completing a task, the format is given by: **[*Agent1 Performing Task* (completed) - *AgentN Performing Task*] *Task***. As more than one person can perform the task, the state of completion is individual to the student.



Smart Jason Agent
[[redacted] (CONCLUÍDO) - [redacted] (CONCLUÍDO) - [redacted] (CONCLUÍDO)] US02 - Task 03 Carregar os dados do usuário na tela perfil 12:22

Smart Jason Agent
US03 - Carregar os dados do perfil no formulário e permitir edição 12:22

Smart Jason Agent
US06 - Task1 Criar os componentes da tela 12:22

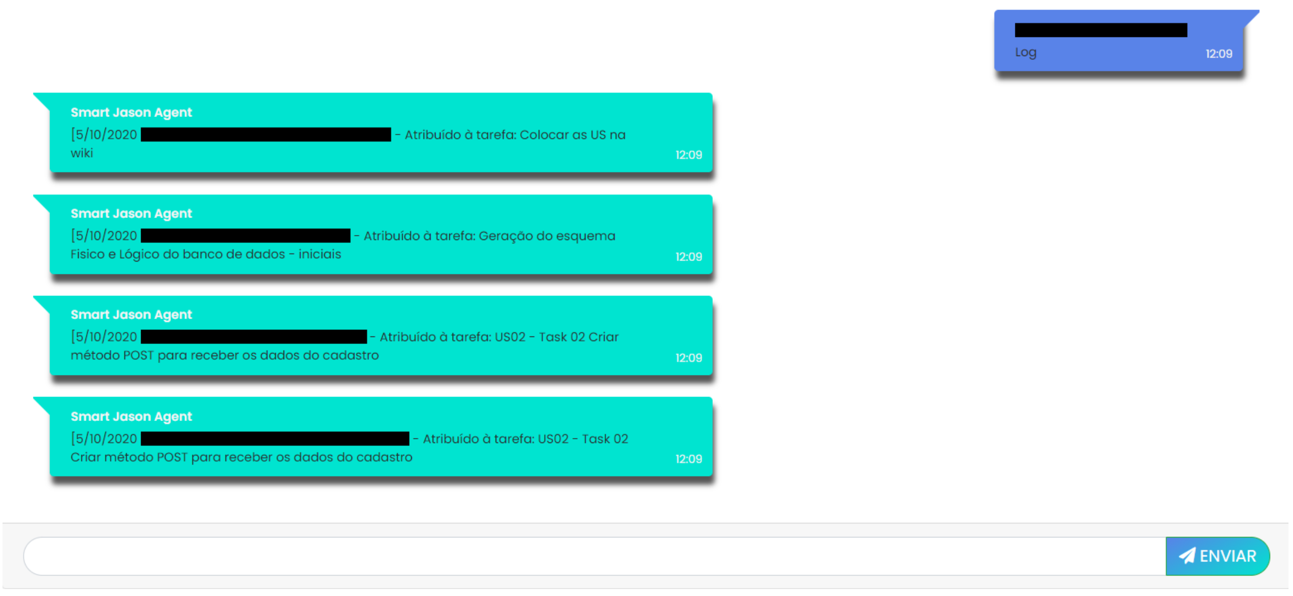
Smart Jason Agent
[[redacted]] US06 - Task2 Criar a tela com os componentes e integrá-los 12:22

Smart Jason Agent
[[redacted]] US06 - Task3 Criar método GET para listar as vagas 12:22

Smart Jason Agent
Adicionar Navigation Bar 12:22

Figure 5.2 – Task List Request

Figures 5.3 and 5.4 shows a request made by a group leader regarding his group's Log. The agent's response is in the format: **[Date of Assignment *Agent*] Assigned task: *Task***.



[redacted]
Log 12:09

Smart Jason Agent
[5/10/2020 [redacted] - Atribuído à tarefa: Colocar as US na wiki 12:09

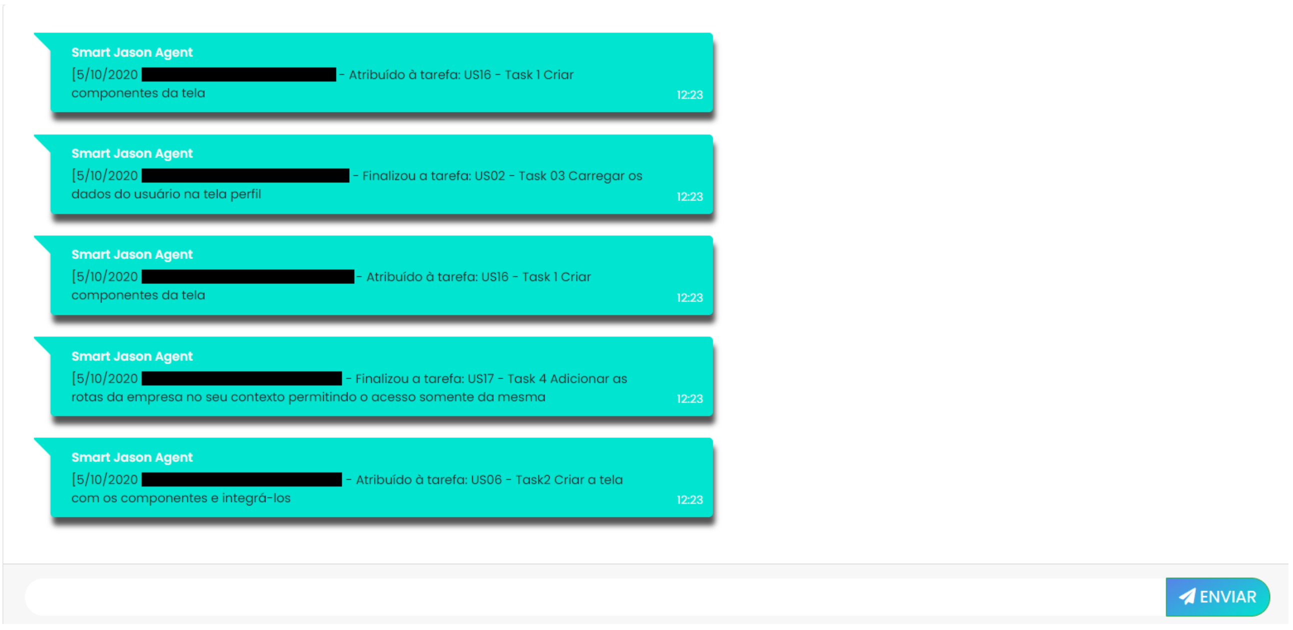
Smart Jason Agent
[5/10/2020 [redacted] - Atribuído à tarefa: Geração do esquema Fisico e Lógico do banco de dados - iniciais 12:09

Smart Jason Agent
[5/10/2020 [redacted] - Atribuído à tarefa: US02 - Task 02 Criar método POST para receber os dados do cadastro 12:09

Smart Jason Agent
[5/10/2020 [redacted] - Atribuído à tarefa: US02 - Task 02 Criar método POST para receber os dados do cadastro 12:09

Figure 5.3 – Log Request

The task completion status is also shown in the log if the agent responsible for executing the task completes the task, as shown in Figure 5.4. In the case of completing a task, the format is given by: **[Date of Assignment *Agent*] Completed the task: *Task***.



Smart Jason Agent
[5/10/2020 [redacted] - Atribuído à tarefa: US16 - Task 1 Criar componentes da tela 12:23

Smart Jason Agent
[5/10/2020 [redacted] - Finalizou a tarefa: US02 - Task 03 Carregar os dados do usuário na tela perfil 12:23

Smart Jason Agent
[5/10/2020 [redacted] - Atribuído à tarefa: US16 - Task 1 Criar componentes da tela 12:23

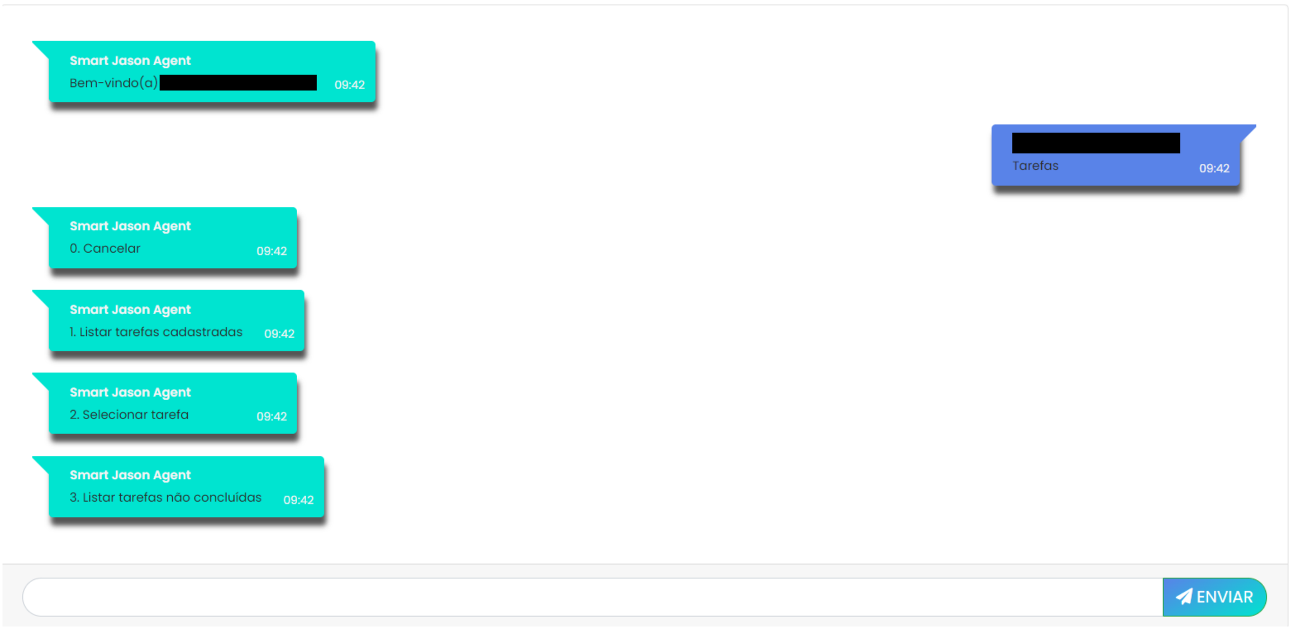
Smart Jason Agent
[5/10/2020 [redacted] - Finalizou a tarefa: US17 - Task 4 Adicionar as rotas da empresa no seu contexto permitindo o acesso somente da mesma 12:23

Smart Jason Agent
[5/10/2020 [redacted] - Atribuído à tarefa: US06 - Task2 Criar a tela com os componentes e integrá-los 12:23

Figure 5.4 – Log Request

Figures 5.5 and 5.6 show examples of requests based on selecting options from the student. These figures are examples of responses when a multi-agent system handles requests that have follow-up intents, the next user interaction, the chatbot will take into account that the request is in the context of *Tasks* (Figure 5.5) and *My Tasks* (Figure 5.6), because follow-up intents keep the context and/or parameters for a certain number of interactions.

Figure 5.5 shows a student requesting the available options regarding tasks. The agent returns the following response: **0. Cancel; 1. List registered tasks; 2. Select task and 3. List unfinished tasks**.



Smart Jason Agent
Bem-vindo(a) [redacted] 09:42

[redacted]
Tarefas 09:42

Smart Jason Agent
0. Cancelar 09:42

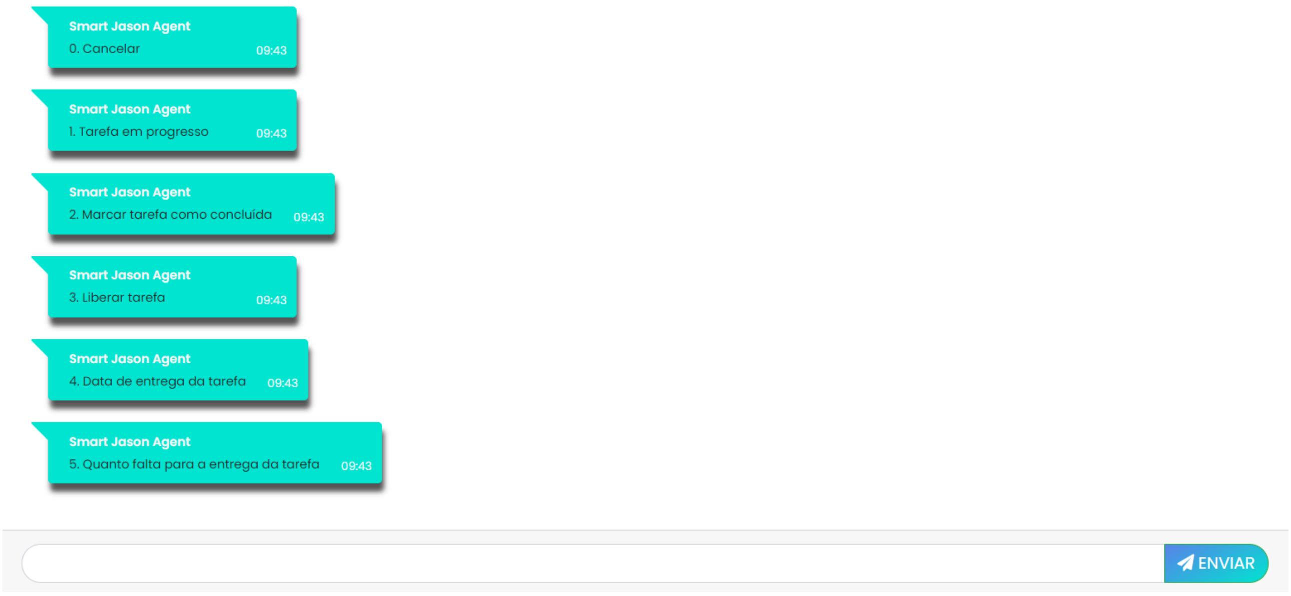
Smart Jason Agent
1. Listar tarefas cadastradas 09:42

Smart Jason Agent
2. Selecionar tarefa 09:42

Smart Jason Agent
3. Listar tarefas não concluídas 09:42

Figure 5.5 – Task Request

Figure 5.6 shows a student requesting the options available regarding the agent's tasks. The agent returns the following response: **0. Cancel; 1. Task in progress; 2. Mark task as completed; 3. Release task; 4. Delivery date of the task and 5. How much is left for the delivery of the task**,



Smart Jason Agent
0. Cancelar 09:43

Smart Jason Agent
1. Tarefa em progresso 09:43

Smart Jason Agent
2. Marcar tarefa como concluída 09:43

Smart Jason Agent
3. Liberar tarefa 09:43

Smart Jason Agent
4. Data de entrega da tarefa 09:43

Smart Jason Agent
5. Quanto falta para a entrega da tarefa 09:43

Figure 5.6 – My Task Request

Two types of logs were generated by the system, one informing about activities and triggering intentions and the other form of validation was through the training guide of

Dialogflow itself (Figure 5.7). The following are examples of the system logs.

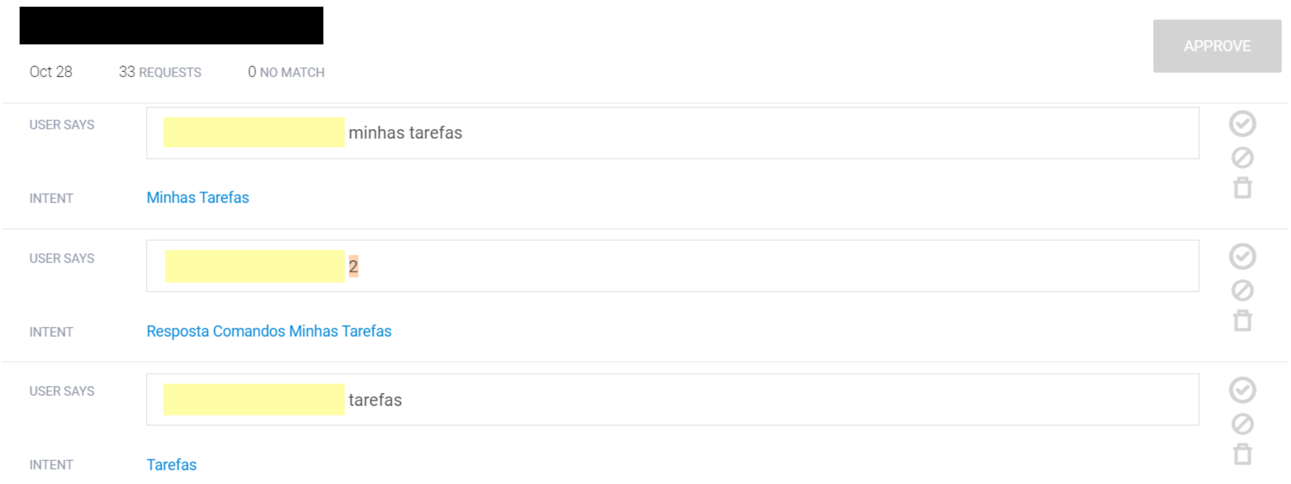
Activities
Agent - Assigned to task: The logout button must be removed from all screens being used and must pass into the 'More ...' tab bar section
Agent - Assigned to task: Update Wiki with Mockups
Agent - Assigned to task: On the job vacancy screen (company) it must contain the fields 'Hours' and 'Local'
Agent - Assigned to task: On the login screen the enter button must be with the same rounding as the first screen
Agent - Assigned to task: Adjust button colours to black, must be the same as the login screen
Agent - Assigned to task: Increase the logo on the quick registration screen
Agent - Assigned to task: US03 Validate the data entered by the user and apply masks
Agent - Assigned to task: US03 Complete address fields via API
Agent - Assigned to task: US03 Load profile data into the form and allow editing
Agent - Assigned to task: US03 Load profile data into the form and allow editing
Agent - Assigned to task: Design Adjust vacancy registration screen according to the current project
Agent - Assigned to task: On the Search for Available Jobs screen, the title should be aligned in the left paragraph format
Agent - Assigned to task: US06 Consume zip code API to fill in data based on zip code
Agent - Assigned to task: Automated tests
Agent - Assigned to task: US16 Create profile screen
Agent - Assigned to task: US16 Create screen components
Agent - Assigned to task: US16 Integrate screen and components
Agent - Assigned to task: US16 Create profile screen
Agent - Assigned to task: US16 Create screen components
Agent - Assigned to task: US16 Integrate screen and components
Agent - Assigned to task: US16 Create profile screen
Agent - Assigned to task: US16 Create screen components
Agent - Assigned to task: US16 Integrate screen and components
Agent - Assigned to task: Create classes to standardise calls to Firebase
Agent - Assigned to task: Add descriptions to photos
Agent - Assigned to task: 33 Delete discipline (and your photos)
Agent - Assigned to task: Import photos from user's phone automatically
Agent - Assigned to task: 39 Optical character recognition (OCR)
Agent - Finished the task: The logout button must be removed from all screens that are being used and must pass into the 'More ...' tab bar section
Agent - Finished the task: Update Wiki with Mockups

Agent - Finished the task: On the login screen the enter button must be with the same rounding as the first screen
Agent - Finished the task: Adjust button colours to black, must be the same as the login screen
Agent - Finished the task: US03 Validate the data entered by the user and apply masks
Agent - Finished the task: US03 Complete address fields via API
Agent - Finished the task: US03 Load profile data in the form and allow editing
Agent - Finished the task: US03 Load profile data in the form and allow editing
Agent - Finished the task: Design Adjust vacancy registration screen according to the current project
Agent - Finished the task: On the Search for Available Jobs screen, the title should be aligned in the left paragraph format
Agent - Finished the task: US06 Consume zip API to fill in data based on zip code
Agent - Finished the task: US16 Create profile screen
Agent - Finished the task: US16 Create screen components
Agent - Finished the task: US16 Integrate screen and components
Agent - Finished the task: US16 Create profile screen
Agent - Finished the task: US16 Create screen components
Agent - Finished the task: US16 Integrate screen and components
Agent - Finished the task: US16 Create profile screen
Agent - Finished the task: US16 Create screen components
Agent - Finished the task: US16 Integrate screen and components

Intents
Received (*Agent*) intent: Create Group
Received (*Agent*) intent: Commands
Received (*Agent*) intent: Response Commands
Received (*Agent*) intent: Task Commands Response
Received (*Agent*) intent: Create Group
Received (*Agent*) intent: Commands
Received (*Agent*) intent: Response Commands
Received (*Agent*) intent: Task Commands Response
Received (*Agent*) intent: Log
Received (*Agent*) intent: Tasks
Received (*Agent*) intent: Task Commands Response
Received (*Agent*) intent: Select Task
Received (*Agent*) intent: My Tasks
Received (*Agent*) intent: My Tasks Commands Response
Received (*Agent*) intent: Group
Received (*Agent*) intent: Group Commands Response

Received (*Agent*) intent: Project
Received (*Agent*) intent: Developed
Received (*Agent*) intent: Difference Delivery Date
Received (*Agent*) intent: Goal

Users' requests for the chatbot are perceived by the Dialogflow platform and stored in the training guide, as shown in Figure 5.7, storing information such as: date of the request, who requested it, what was their input, what intention was triggered and what parameters were extracted. Figure 5.7 shows an example of Dialogflow training, in which the intention that was triggered and the parameters recognised are shown, in yellow the `name` parameter and in orange the `operation` of a command request.


[redacted]

Oct 28 33 REQUESTS 0 NO MATCH

USER SAYS minhas tarefas
INTENT Minhas Tarefas

USER SAYS 2
INTENT Resposta Comandos Minhas Tarefas

USER SAYS tarefas
INTENT Tarefas

Figure 5.7 – Dialogflow training example

6. FINAL CONSIDERATIONS AND FUTURE WORK

Increasingly, remote (non-face-to-face) learning has been used in the education or training of people through digital resources to acquire new knowledge, to develop professionally by acquiring new skills and abilities of the most diverse types, among others. With reasonable technological support, distance education is being used as an alternative to face-to-face activities to continue education amid the restrictions imposed by the pandemics of COVID-19.

In this context, distance education allows people who do not have access to information in physical environments (for social reasons or for a specific situation, such as the pandemic) to easily, quickly and dynamically consume personalised and efficient knowledge from a digital platform. Although distance learning has so many good points, it is still far from being the ideal method. There is a concern about the collaborative distance learning method, in which the main problems to be solved in this approach are: balance of skills within a group, incorrect group dynamics, lack of communication in the group and difficulty with social situations.

As demonstrated in this work, Allaymoun and Trausan-Matu [2], Ferschke, Tomar and Rosé [20] among other authors showed that collaborative work is an important factor in a student's development. In addition, Ferschke, Tomar and Rosé [20] shows the importance of avoiding redundant work by sharing information within a group. Thus, this work proposed an approach through multi-agent systems, which differs from other works covered in this dissertation, in which the greatest means of disseminating information was a large information store mediated by a chatbot. Using a multi-agent system, it was possible to achieve approaches by organisations (groups), in which, unlike other approaches, our system is capable of managing multiple groups, managing information such as roles and tasks, in which each group has its own organisation configuration, that is, their own roles, tasks and restrictions.

Therefore, the main objective of the multi-agent system is to manage the organisation of a real group through its representation in the system, by facilitating the dissemination of information about the current status of the project and allowing the responsible for the group to monitor the performance of the members during the tasks of execution, without disturbing the construction of knowledge. In particular, the focus is on the educational environment, where it is very important that project members are aware of how the project is evolving. An approach based on multi-agent systems is used, specifically with the JaCaMo platform, because it is possible to create and control an organisation in terms of members and in the management of task assignment. Communication with the user through a chatbot aims to allow a more effective and natural communication with the system.

The possibility of creating multiple domains is what makes the use of multi-agent systems a very important feature for managing systems. In particular, if there are different people or organisations with different objectives and proprietary information, a multi-agent system is capable of handling interactions efficiently. That is, MAS are able to model an organisation's internal affairs with a single system, avoiding the need to develop an organisation that encompasses all representations, but rather to develop different organisations that are accessible in their own system with their capabilities and priorities.

It was noticed during the development of this work that multi-agent systems present a range of possibilities in the coordination of groups in an educational environment, enabling group management through tasks or roles. Multi-agent systems are able to facilitate the development of multiple projects simultaneously due to the selection of tasks and roles necessary to create organisations in the system, allow restrictions on the execution of tasks based on the capacity (role) of the agent, allow group members to be attentive to all the advances that occur in the development of the project through the Moise management, among other possibilities.

The biggest challenge of this work when implementing a multi-agent system in a collaborative educational project is to improve the system so that changes in the group structure occur during the development of the project, that is, change the Moise scheme at run-time to insert new ones plans and consequently missions without interruption in the system.

As a future work, we intend to make the use of chatbot more flexible, allowing a different approach in the system when carrying out interventions in the group with proactivity, change the chat interface so that students can communicate through the system interface itself, allow a means of direct communication with the teacher through the platform, apply and test the method of the team composition problem, previously interview users to collect the phrases used in the development of a work to enrich the chatbot language and finally, researching ways to change the group's schema at run-time so that the group leader can change through the proposed interface without interruptions in the system.

As a future work, we intend to make the use of the chatbot more flexible, allowing a proactive approach to the system when carrying out interventions in the group; change the chat interface so that students can communicate through the system interface itself; allow a means of direct communication with the teacher through the platform; apply and test the method presented in Section 2.5 to solve the team composition problem; interview the users to collect the phrases used in the development of a groupwork to enrich the chatbot language; and finally, research ways to change the scheme of the group at run-time so that the group leader can change through the proposed interface without interruptions in the system.

REFERENCES

[1] Al-Abri, A.; AlKhanjari, Z.; Jamoussi, Y.; Kraiem, N. “Identifying learning styles from chat conversation using ontology-based dynamic bayesian network model”. In: Proceedings of the 8th International Conference on Computer Science and Information Technology, 2018, pp. 77–84.

[2] Allaymoun, M. H.; Trausan-Matu, S. “Analysis of collaboration in computer supported collaborative learning chat using rhetorical schemas”. In: Proceedings of the 7th International Conference on Information and Communication Systems, 2016, pp. 39–44.

[3] Andrejczuk, E.; Berger, R.; Rodríguez-Aguilar, J. A.; Sierra, C.; Marín-Puchades, V. “The composition and formation of effective teams: computer science meets organizational psychology”, *The Knowledge Engineering Review*, vol. 33, Nov 2018, pp. e17.

[4] Andrejczuk, E.; Rodríguez-Aguilar, J. A.; Roig, C.; Sierra, C. “Synergistic team composition”. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, 2017, pp. 1463–1465.

[5] Baby, C. J.; Khan, F. A.; Swathi, J. N. “Home automation using iot and a chatbot using natural language processing”. In: Proceedings of the Institute of Electrical and Electronics Engineers Innovations in Power and Advanced Computing Technologies, 2017, pp. 1–6.

[6] Baylor, A. L.; Kim, Y. “Simulating instructional roles through pedagogical agents”, *International Journal of Artificial Intelligence in Education*, vol. 15–2, Jan 2005, pp. 95–115.

[7] Belbin, R. M. “Team roles at work”. Routledge, 2012, 162p.

[8] Boissier, O.; Bordini, R. H.; Hübner, J. F.; Ricci, A.; Santi, A. “Multi-agent oriented programming with jacamo”, *Science of Computer Programming*, vol. 78–2, Jun 2013, pp. 747–761.

[9] Bonk, C.; Zhang, K. “Introducing the r2d2 model: online learning for the diverse learners of this world”, *Distanec Education*, vol. 27–2, Aug 2006, pp. 249–264.

[10] Bordini, R. H.; Dastani, M.; Dix, J.; Fallah-Seghrouchni, A. E. “Multi-agent programming, languages, tools and applications”. Springer, 2009, 420p.

[11] Bordini, R. H.; Hübner, J. F.; Wooldridge, M. “Programming multi-agent systems in AgentSpeak using Jason”. John Wiley & Sons, 2007, 292p.

[12] Brazier, F. M. T.; Mobach, D. G. A.; Overeinder, B. J.; Wijngaards, N. “Supporting life cycle coordination in open agent systems”. In: Proceedings of the Multiagent System Problem Spaces Workshop at Autonomous Agents and Multiagent Systems, 2002, pp. 1–4.

[13] Capone, C.; Bordini, R. H.; Mascardi, V.; Delzanno, G.; Ferrando, A.; Gelati, L.; Guerrini, G. “Smart rogagent: where agents and humans team up”. In: Proceedings of the 22nd Principles and Practice of Multi-Agent Systems, 2019, pp. 541–549.

[14] Chowdhury, G. G. “Natural language processing”. John Wiley & Sons, 2003, chap. 2, pp. 51–89.

[15] David, B.; Chalon, R.; Zhang, B.; Yin, C. “Design of a collaborative learning environment integrating emotions and virtual assistants (chatbots)”. In: Proceedings of the 23rd Institute of Electrical and Electronics Engineers International Conference on Computer Supported Cooperative Work in Design, 2019, pp. 51–56.

[16] Díaz, L. A.; Entonado, F. B. “Are the functions of teachers in e-learning and face-to-face learning environments really different?”, *Educational Technology & Society*, vol. 12–4, Oct 2009, pp. 331–343.

[17] Dutta, D. “Developing an intelligent chat-bot tool to assist high school students for learning general knowledge subjects”, Technical Report, Georgia Institute of Technology, 2017, 13p.

[18] Er, E.; Ozden, M.; Arifoglu, A. “Livelms: a blended e-learning environment: a model proposition for integration of asynchronous and synchronous e-learning”, *International Journal of Learning*, vol. 16–2, Jan 2009, pp. 449–460.

[19] Farhangian, M.; Purvis, M. K.; Purvis, M.; Savarimuthu, B. T. R. “Agent-based modeling of resource allocation in software projects based on personality and skill”. In: Proceedings of the Advances in Social Computing and Multiagent Systems, 2015, pp. 130–146.

[20] Ferschke, O.; Tomar, G.; Rosé, C. P. “Adapting collaborative chat for massive open online courses: lessons learned”. In: Proceedings of the 17th International Conference on Artificial Intelligence in Education, 2015, pp. 13–18.

[21] Garrison, D.; Kanuka, H. “Blended learning: uncovering its transformative potential in higher education”, *The Internet and Higher Education*, vol. 7–2, Apr 2004, pp. 95–105.

[22] Göschlberger, B.; Brandstetter, C. “Conversational AI for corporate e-learning”. In: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, 2019, pp. 674–678.

[23] Guo, P. J.; White, J.; Zanelatto, R. “Codechella: multi-user program visualizations for real-time tutoring and collaborative learning”. In: Proceedings of the Institute of Electrical and Electronics Engineers Symposium on Visual Languages and Human-Centric Computing, 2015, pp. 79–87.

[24] Horling, B.; Lesser, V. R. “A survey of multi-agent organizational paradigms”, *The Knowledge Engineering Review*, vol. 19, Dec 2004, pp. 281–316.

[25] Hrastinski, S. “Asynchronous and synchronous e-learning”, *Educause Quarterly*, vol. 31–4, Jan 2008, pp. 51–55.

[26] Hübner, J. F.; Boissier, O.; Kitio, R.; Ricci, A. “Instrumenting multi-agent organisations with organisational artifacts and agents”, *Autonomous Agents and Multi-Agent Systems*, vol. 20, May 2010, pp. 369–400.

[27] Hübner, J. F.; Sichman, J. S.; Boissier, O. “Developing organised multiagent systems using the moise$^+$ model: programming issues at the system and agent levels”, *International Journal of Agent-Oriented Software Engineering*, vol. 1, Dec 2007, pp. 370–395.

[28] Huhns, M. N.; Stephens, L. M. “Multiagent systems and societies of agents”. Massachusetts Institute of Technology Press, 1999, chap. 2, pp. 79–120.

[29] Islam, A.; Flint, J.; Jaecks, P.; Cap, C. “A proficient and versatile online student-teacher collaboration platform for large classroom lectures”, *International Journal of Educational Technology in Higher Education*, vol. 14–1, Nov 2017, pp. 29.

[30] Jung, C.; Hull, R. “Collected works of CG Jung, volume 6: Psychological types”. Princeton University Press, 1976, 640p.

[31] Kane, D. A. “The role of chatbots in teaching and learning”. McFarland, 2016, chap. 4, pp. 131–148.

[32] Kerly, A.; Ellis, R.; Bull, S. “Conversational agents in e-learning”. In: Proceedings of the XVI Applications and Innovations in Intelligent Systems, 2008, pp. 169–182.

[33] Kim, S.; Eun, J.; Oh, C.; Suh, B.; Lee, J. “Bot in the bunch: facilitating group chat discussion by improving efficiency and participation with a chatbot”. In: Proceedings of the 20st Conference on Human Factors in Computing Systems, 2020, pp. 1–13.

[34] King, A. “Structuring peer interaction to promote high-level cognitive processing”, *Theory Into Practice*, vol. 41, Mar 2002, pp. 33–39.

[35] Kitchenham, B. “Procedures for performing systematic reviews”, Technical Report, Keele University, 2004, 33p.

[36] Kozslowski, S.; Bell, B. “Work groups and teams in organizations. Review update”. John Wiley & Sons, 2012, chap. 3, pp. 412–470.

[37] Krausburg, T. “Constrained coalition formation among heterogeneous agents for the multi-agent programming contest”, Masters Dissertation, Computer Science Graduate Program, School of Technology, Pontifical Catholic University of Rio Grande do Sul, 2018, 109p.

[38] Kumar, R.; Beuth, J.; Rosé, C. P. “Conversational strategies that support idea generation productivity in groups”. In: Proceedings of the 9th Computer Supported Collaborative Learning, 2011, pp. 398–406.

[39] Kumar, R.; Rosé, C. P. “Architecture for building conversational agents that support collaborative learning”, *Institute of Electrical and Electronics Engineers Transactions on Learning Technologies*, vol. 4–1, Jan-Mar 2011, pp. 21–34.

[40] Lorenzo, G.; Ittelson, J. “An overview of e-portfolios”, *Educause Learning Initiative*, vol. 1–1, Jan 2005, pp. 1–27.

[41] Martínez-Caro, E. “Factors affecting effectiveness in e-learning: an analysis in production management courses”, *Computer Applications in Engineering Education*, vol. 19–3, Sep 2011, pp. 572–581.

[42] Miller, G. A. “Wordnet: a lexical database for english”, *Communications of the Association for Computing Machinery*, vol. 38–11, Nov 1995, pp. 39–41.

[43] Myers, I. B.; Myers, P. B. “Gifts differing: Understanding personality type”. Nicholas Brealey, 2010, 248p.

[44] Neto, A. J. M.; Fernandes, M. A. “Chatbot and conversational analysis to promote collaborative learning in distance education”. In: Proceedings of the 19th Institute of Electrical and Electronics Engineers International Conference on Advanced Learning Technologies, 2019, pp. 324–326.

[45] Paikari, E.; Choi, J.; Kim, S.; Baek, S.; Kim, M.; Lee, S.; Han, C.; Kim, Y.; Ahn, K.; Cheong, C.; van der Hoek, A. “A chatbot for conflict detection and resolution”. In: Proceedings of the 1st International Workshop on Bots in Software Engineering, 2019, pp. 29–33.

[46] Pham, X. L.; Pham, T.; Nguyen, Q. M.; Nguyen, T. H.; Cao, T. T. H. “Chatbot as an intelligent personal assistant for mobile language learning”. In: Proceedings of the 2nd International Conference on Education and E-Learning, 2018, pp. 16–21.

[47] Ramezan, M. “Intellectual capital and organizational organic structure in knowledge society: How are these concepts related?”, *International Journal of Information Management*, vol. 31, Feb 2011, pp. 88–95.

[48] Sandholm, T.; Larson, K.; Andersson, M.; Shehory, O.; Tohmé, F. "Coalition structure generation with worst case guarantees", *Artificial Intelligence*, vol. 111–1–2, Jul 1999, pp. 209–238.

[49] Tegos, S.; Demetriadis, S. N. "Conversational agents improve peer learning through building on prior knowledge", *Educational Technology & Society*, vol. 20–1, Jan 2017, pp. 99–111.

[50] Tisue, S. "Netlogo: design and implementation of a multi-agent modeling environment". In: Proceedings of the Agent Conference on Social Dynamics: Interaction, Reflexivity and Emergence, 2004, pp. 161–184.

[51] Trausan-Matu, S. "A polyphonic model, analysis method and computer support tools for the analysis of socially-built discourse", *Romanian Journal of Information Science and Technology*, vol. 16–2–3, Jan 2013, pp. 144–154.

[52] Veletsianos, G.; Russell, G. S. "Pedagogical agents". Springer, 2014, chap. 7, pp. 759–769.

[53] Vrajitoru, D.; Ratkiewicz, J. "Evolutionary sentence combination for chatterbots". In: Proceedings of the International Conference on Artificial Intelligence and Applications, 2004, pp. 287–292.

[54] Weyns, D.; Omicini, A.; Odell, J. "Environment as a first class abstraction in multiagent systems", *Autonomous Agents and Multi-Agent Systems*, vol. 14, Feb 2007, pp. 5–30.

[55] Wilde, D. "Post-Jungian personality theory for individuals and teams". Sydrose LP, 2013, 23p.

[56] Williams, J. D.; Kamal, E.; Ashour, M.; Amr, H.; Miller, J.; Zweig, G. "Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service". In: Proceedings of the 16th Special Interest Group on Discourse and Dialogue, 2015, pp. 159–161.

[57] Wooldridge, M. J. "An introduction to multiagent systems, second edition". John Wiley & Sons, 2009, 461p.

APPENDIX A – GROUP TASKS

Tasks Group 1 – Tasks that students had to solve were related to software development, following the list of tasks:

User story planning
Mockup planning
Study Flutter / Dart and start thinking about architecture
Study Firebas and think about textual search, image processing and storage

See photos organised by discipline
Register schedule grid
Create the logical model of the application data
Setup of the Flutter project
Adapt UX according to stakeholder feedbacks

Search for the description of photos of a course
Adjust communication with back-end on the creation screen
Adjust times and colours
Delete photos

Fix navigation bug when deleting photos
Add arrows indicating whether the course is collapsed or expanded

Remove button from general search, leave only in the scope of the course and remove Tags
Automatically import photos from user's phone
Edit course
View photos on any device
Install APK and generate a list of bugs
Review photo sharing
Optical character recognition (OCR)
Delete grid (and its disciplines)
Add description to photos
Delete subjects (and your photos)
Replace app mocks with real calls
Fix infinite bug loading after course swipe
Create class to standardize Firebase calls
Share photos with colleagues

Bug on the screen of the list of photos in which some images are staying on top of others
Correct the title of the photo listing screen

Download APK and map bugs
Log out of settings
Discipline does not appear after being added Necessary to go to another screen and return
Fix app on iOS. Path of hard coded files may be causing the problem

Remove unnecessary calls to Firebase
Edit Grid
Add description to specific areas of the photos

Display dates, day of the week and time instead of the path in synced photos
Descending semesters in a decreasing way
Fix course collapse
Validate that the description is not being persisted after changing Visual feedback on registration
Review the schedule format / validate if it is from the simulator and remove from the American standard
Callout on the sync tab
Hide bar on login screen
Subscribe to the wiki list of known issues
Review application texts
Review login

Tasks Group 2 – Tasks that students had to solve were related to software development, following the list of tasks:

Initial setup
Cancel vacancy advertised
Add button to log out and remove vacancy in the back-end
In the edit profile screen, the shift field must be readjusted to the same as the photo, adding identity
Move password encoding to the back-end

Make sure only applied vacancies are shown when clicking on applications

Reflect the edit data to the profile view screen
Test development
In the screen of registering vacancies (company) it must contain the fields "Schedules" and

"Location"
Automated testing
Create a job posting screen
Bug of the company registration screen does not appear
Create company vacancies page redirect button to view company profile

Update mockups as described
Update the wiki with the photo of team members
Register in the application to publish vacancies
Show registered jobs
Update tab of sprints with the US that were delivered and technical charge
Create installation script for the user to download the project and run
Add wiki configuration section
Edit the profile through the home screen for easy access
Merge the edit profile screen with dev
Create registration screen with respective fields
Update missing field on back-end
Search for company data to store in user data and add the "company name" field to the company table
When a skill or experience is inserted, when the data is saved, it must be displayed within a modal
Skills and experiences fields should be formatted
Fix login screen responsiveness
Navigate the platform without having registered to view vacancies
In my skills and experience section the data must be displayed inside a card
In the editor screen the profile, the duration fields must be changed to start date and end date
Edit profile
Increase the logo on the quick registration screen
Validate edit fields
Create company profile edit screen
Create components of the company profile edit screen
Frontend and back-end integration
Create endpoint patch to edit company
Bugfix registration screen is not working correctly, it is not possible to perform a registration
Fix the screen to search for available vacancies, the title should be aligned in the paragraph format on the left
Fix on the login screen the button to enter must be with the same rounding as the first screen
Fix adjust button colours to black, should be the same as the login screen Feature-log out the log out button must be removed from all screens being used, must pass into the header

Company can access the profile
Integrate front-end and back-end and open US PR
Create profile screen
Create screen components
Integrate screen and components
Adjust vacancy screen according to the current project
Update wiki with mockups
Company can register vacancy in the application
Add the company's routes in their context allowing access only from the same
Create screen components
Create POST method for data insertion
Create the screen with the components
User view the job details of interest
Create the screen with the components and integrate them
Create get method to list vacancies
Consume CEP API to fill data based on CEP
Create the screen components
Create controller and DAO for data editing
Create route with PUT method in the API to edit the data
Complete address fields via API
Load profile data into the edit form
Validate the data entered by the user and apply masks
User view profile for data verification
Load user data on the profile screen
Create profile screen
Create screen components
Create POST method to receive registration data
Create GET endpoint in the users API
Allow user registration in the application
Create input component
Create user registration route
Back-end and front-end integration
Capture the data entered in the fields
Ensure responsiveness of the elements
Create quick registration screen
Create BD model to persist registration data
Create button component
Create home screen button routes
Create home screen
Navigation Buttons

Button Profile
Review login flow when registering
Add splash screen for PWA
Make mockups
Adjust mockups as per customer request
User route management
Initial back-end structure
Generation of the physical and logical schema of the database
Create back-end API documentation (SWAGGER)
Update the wiki with the charter
Put the US on the wiki
Initial front-end structure
Add the initial versions of the database model to the wiki
Component and deploy diagrams
Refactor front-end structure

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br